NONNEGATIVE MATRIX FACTORIZATION FOR TEXT, GRAPH, AND HYBRID DATA ANALYTICS

A Dissertation
Presented to
The Academic Faculty

By

Rundong Du

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mathematics

Georgia Institute of Technology

May 2018

NONNEGATIVE MATRIX FACTORIZATION FOR TEXT, GRAPH, AND HYBRID DATA ANALYTICS

Approved by:

Dr. Haesun Park, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Duen Horng Chau
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Edmond Chow
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Sung Ha Kang
School of Mathematics
Georgia Institute of Technology

Dr. Hao-Min Zhou
School of Mathematics
Georgia Institute of Technology

Date Approved: March 12, 2018

To my parents.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor Prof. Haesun Park. She introduced me to the world of nonnegative matrix factorization (NMF) and with her guidance I am able to appreciate the simpleness, the beauty and the power of NMF based methods. She taught me how to do research and think independently, she guided me to interesting and useful research directions and she provided insightful advises that helped me out when I encountered obstacles in my research. She provided me many opportunities to collaborate with other great researchers and participate in many interesting real-world projects. Besides research, she also provided much valuable guidance on my writing, my career and general academic life.

I would like to thank my committee members Prof. Duen Horng Chau, Prof. Edmond Chow, Prof. Sung Ha Kang and Prof. Hao-Min Zhou for their time, service and their valuable comments and suggestions. In addition, I would like to thank Prof. Sung Ha Kang, who is also my mentor in the School of Mathematics, for her kind and helpful life and academic tips. I'd like to thank Prof. Bistra Dilkina, who served on my proposal committee before she left Georgia Tech.

I would like to thank Da Kuang, who was a senior PhD student when I entered the group. He kindly provided me much hands on coaching on doing research, writing papers and coding efficient implementations of NMF algorithms. We had many great collaborations. I thank Barry Drake for many inspiring discussions, his kind help in writing, his contributions to many papers we co-authored and I thank his team in Georgia Tech Research Institute, especially Ashley Beavers and Tiffany Huang, for making efficient implementations of NMF algorithms and data processing tools with great quality, which I depended on a lot in my research. I also thank Ashley Beavers for a productive collaboration in a Hackathon. I thank Woosang Lim for many enjoyable and deep discussions, exciting collaborations on many interesting topics, and his hard working in our co-authored work. I thank Zhongming Lu for interesting collaborations on sustainability. I thank Yunlong He for the proof of Theorem 3.1. I also would like to thank Jaegul Choo, Ramakrishnan Kannan, Hannah Kim, Seungyeon Kim, Bo Xie, Jun Lu, Tongzhou Chen and Yichen Wang for the helpful discussions and exciting moments.

I would like to thank all the nice faculty and staff I have met in the School of Computational Science and Engineering, and the School of Mathematics. I appreciate their kind help to me and their warm greetings. They have made my PhD life in Georgia Tech smooth and happy.

Finally, I would like to thank my greatest wife, Qingqing Liu, for her love, support, encouragement and for bringing our lovely daughter into the world. They are the source of my strength and happiness. It is impossible for me to finish the thesis on time without her thoughtful and selfless help in life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Constrained low rank approximation is a general framework for data analysis, which usually has the advantage of being simple, fast, scalable and domain general. One of the most known constrained low rank approximation methods is nonnegative matrix factorization (NMF). This research studies the design and implementation of several variants of NMF for text, graph and hybrid data analytics. It will address challenges including solving new data analytics problems and improving the scalability of existing NMF algorithms.

There are two major types of matrix representation of data: feature-data matrix and similarity matrix. Previous work showed successful application of standard NMF for feature-data matrix to areas such as text mining and image analysis, and Symmetric NMF (SymNMF) for similarity matrix to areas such as graph clustering and community detection. In this work, a divide-and-conquer strategy is applied to both methods to improve their time complexity from cubic growth with respect to the reduced low rank to linear growth, resulting in DC-NMF and HierSymNMF2 methods. Extensive experiments on large scale real world data show improved performance of these two methods.

Furthermore, in this work NMF and SymNMF are combined into one formulation called JointNMF, to analyze hybrid data that contains both text content and connection structure information. Typical hybrid data where JointNMF can be applied includes paper/patent data where there are citation connections among content and email data where the sender/receipts relation is represented by a hypergraph and the email content is associated with hypergraph edges. An additional capability of the JointNMF is prediction of unknown network information which is illustrated using several real world problems such as citation recommendations of papers and activity/leader detection in organizations.

This dissertation also includes brief discussions of relationship among different variants of NMF.

CHAPTER 1
INTRODUCTION AND BACKGROUND

In the era of information explosion, the amount of data and information has been quickly growing to a level that in many situations, data analytics and information retrieval can not be done without the help of computer algorithms. Advanced, AI-oriented algorithms have been developed to process and understand the data. For example, to analyze texts, natural language models have been built to analyze grammars and syntax of sentences, and cognitive models were developed to make inferences based on the language structure. Those advanced models and methods are usually carefully designed with complicated assumptions/rules for a specific domain. They are powerful but sometimes are overkill for some data analytics tasks. No matter what underlying model is behind it, much data can be summarized by some hidden patterns with much lower complexity. For example, all the sentences in an English encyclopedia share the same set of grammar rules, a long article can be categorized by a few topic words, a complicated image reduced to 256 colors is still identifiable by a human, etc. Constrained low rank approximation is a category of methods that try to find out the low-complexity patterns behind matrix/tensor encodable data, without complicated assumptions about the model behind the data. In many situations, the hidden patterns discovered by low rank approximation methods give us enough valuable information about the data. Due to the simplicity and independence of the underlying model, low rank approximation methods are widely applicable and easier to scale. For example, nonnegative matrix factorization (NMF), a low rank approximation method with nonnegative constraints, has been applied to document clustering [1], image analysis [2], cancer subtype detection [3], blind source separation for audio [4], and many other areas. Some advantages of NMF based algorithms are: (1) good interpretability (For example, NMF based methods not only give clustering assignments, the generated nonnegative basis vectors also summarize each cluster very well.), (2) being supported by fast numerical routines and sophisticated numerical libraries, and (3) ability to utilize scalable MPI based implementations [5, 6]. This dissertation focuses on NMF and its variants for clustering of text, graph and hybrid data where text and graph are merged.

1.1 Basic Concepts of NMF

Assume the data has nonnegative matrix/vector representation $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}_+^{m \times n}$, where $\mathbb{R}_+$ is the set of all nonnegative real numbers. Nonnegative matrix factorization tries to approximate X as a conic combination of k nonnegative basis vectors, where usually $k \ll \min\{m, n\}$. We assume the k basis vectors are stored in a matrix $W = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k] \in \mathbb{R}^{m \times k}$ and the nonnegative combination coefficients are stored in a matrix $H = (h_{ij}) \in \mathbb{R}^{k \times n}$ such that each $\boldsymbol{x}_j$ is approximated by $\sum_{i=1}^{k} h_{ij} \boldsymbol{w}_i$, or equivalently, the matrix X is approximated by the product WH. Intuitively, if X is well approximated, the column vectors of W will be good representative vectors of the entire data set, and the coefficients in H will show the proportion of each components in each data item. When we have enough representative vectors, most data items will be mainly associated with one representative vector, and therefore a clustering assignment can be induced. More specifically, data $\boldsymbol{x}_j$ belongs to cluster i if $h_{ij} > h_{lj}$ for all $l \neq i$ and $\boldsymbol{w}_i$ is the cluster representative of cluster i. To ensure the quality of such clustering and representative vectors, we would want the approximation error to be as small as possible. The two most common error measures for NMF are Frobenius norm and “generalized”[1] Kullback-Leibler divergence [7]. In this dissertation we only use Frobenius norm which defines NMF as the optimization problem in Equation (1.1)

$$\min_{W \geq 0, H \geq 0} \|X - WH\|_F, \tag{1.1}$$

[1]It is called generalized Kullback-Leibler divergence because here it does not require the input to be probability distribution.

because it has the following advantages:

- Flexibility for designing efficient and scalable algorithms for large-scale problems.
- Ability to produce more accurate solutions in a variety of noisy real-life applications even when other measures such as KL-divergence can model the problems better theoretically [8, 9, 10, 11, 12, 13].
- Convenience to combine low rank approximations of multiple matrices and to add certain regularization terms.

We will illustrate some of these advantages in later chapters.

1.2 Clustering of Text and Graph

Many types of data can be encoded as nonnegative matrix, such as text, graph, image, hyperspectral image, sound, etc. Although NMF does have applications on all those types of data, this dissertation will focus on text and graph data.

1.2.1 Feature-Based Clustering and Similarity-Based Clustering

There are two major types of matrix representation of data: feature-data matrix and similarity matrix. A feature-data matrix is just like matrix X as discussed in Section 1.1. Typically, text data are usually encoded as feature-data matrix, where the features are usually a subset of words that appears in the data set, possibly with some transformations such as stop-word removal, stemming, lemmatization, etc. The most straightforward encoding is perhaps term-frequency matrix, where each entry X_{ij} are integers representing the number of appearance of the i-th word in the j-th document. For better clustering quality, one usually needs to apply some normalizations such as TF-IDF [14] and column normalization. Clustering using a feature-data matrix is called feature based clustering, for which the most famous example is K-means [15]. The standard NMF (1.1) is also a good clustering algorithm for nonnegative feature-data matrices. Particularly, it is shown to be an effective method for text clustering and topic modeling [10]. In Chapter 3, we will develop a fast NMF algorithm called DC-NMF based on a divide-and-conquer strategy.

Sometimes, we only know the relation between data items in a space without knowing their actual vector representation. For example, for kernel methods we only know the inner product or data items in the kernel space; for graph data, the only given information is the connection relation between data items. These relation info can usually be encoded in a symmetric similarity matrix $S \in \mathbb{R}^{n \times n}$, where S_{ij} measures the "strength" of the relation between the i-th and j-th data item. One of the most important types of data that can be represented by a nonnegative symmetric matrix is graph. Besides abundant real-world graphs such as social networks, citation networks, web-linkage networks and communication networks, there are also many graphs designed to model certain relations such as product co-purchasing network, co-author network, etc. In some situations, such as when data points are embedded in a nonlinear manifold, it is better to transform featured-based data into a graph (for example, using k-nearest neighbors) and perform graph clustering. One of the famous graph clustering algorithms is the spectral clustering algorithm [16]. As we will see in Chapter 4, a variant of NMF called Symmetric NMF (SymNMF) is also a good graph clustering algorithm, and again a divide-and-conquer strategy can be applied to make it a large-scale community detection algorithm.

1.2.2 Different Names of Clustering

In different context, clustering may have different names, probably with some subtle differences. Text clustering algorithms and topic modeling methods are largely overlapped, since a cluster representative vector reveals the topic keywords and topic modeling methods assign each document with topics. Topic modeling method can label a document as a weighted mixture of different topics. When it comes to clustering, this is called soft clustering, where a document can belong to multiple clusters with different weights.

In the context of graph clustering, there are names such as community detection and graph partitioning. They are three closely related concepts developed for different goals. Despite their differences in motivations, emphases, and formulations, they all attempt to find a collection of cohesive subsets of nodes in a graph. Community detection seeks to discover subsets of nodes that share some common properties based on network structure. A node may have zero, one, or multiple community affiliations. Graph partitioning problems arise from high performance computing, which can be applied to reordering of sparse matrices and load balancing. Graph partitioning has a strong requirement that every node should belong to one and only one partition, representing *hard* clustering. Graph partitioning usually aims at finding node clusters with equal size and minimizing edge cut. Graph clustering is a more general concept that also includes clustering graph-encoded data from vector spaces. In some work, community detection and graph partitioning are treated as special cases of graph clustering [17]. In fact, one can also use graph partitioning algorithms and graph clustering algorithms for community detection. Community detection and graph clustering have similar goals and are less restrictive than graph partitioning. In this thesis, we do not differentiate community detection and graph clustering.

1.2.3 Evaluation of Clustering Results

There are two types of measures for clustering quality: internal measures and external measures. An internal measure requires the data and the clustering result to compute, while an external measure typically need the clustering result and ground truth clusters. An internal measure is usually a score designed to quantify certain quality of clusters, which largely depends on the type of data. For example, normalized cut, an internal measure for graph clusters, which follows the belief that good graph clusters should have more intra-connections than inter-connections, does not apply to text clusters. Since the objective function of a clustering method is usually believed to describe certain quality that good clusters should have, such objective function is usually used as an internal measure. On the other hand, external measures can usually be applied regardless of the type of data.

For feature-based data, internal measures usually have biased assumptions about the clusters (such as convexity) and it is argued that "good scores on an internal criterion do not necessarily translate into good effectiveness in an application" [14]. For graph data, however, some internal measures are used due to the more unified structure of graphs. Yang and Leskovec [18] studies 13 internal measures and apply them on several real-world networks with ground truth communities, concluding that certain internal measures can capture ground truth clusters very well. These internal measures usually try to capture the concept that good communities has more intra-connections than inter-connections. One of the best internal measures in their study is called conductance, the per community average of which is equivalent to the average normalized cut defined in the next section.

Average Normalized Cut—An Internal measure for Community Detection

Suppose we have a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the weight of an edge (u, v) is $w(u, v)$. Note that for an unweighted graph, $w(u, v) = 1$ if edge $(u, v) \in \mathcal{E}$, otherwise $w(u, v) = 0$. Let $P_1, \ldots, P_k$ be k pair-wise disjoint subsets of $\mathcal{V}$, where $\bigcup_{i=1}^{k} P_i = \mathcal{V}$, then the normalized cut of the partition $(P_1, \ldots, P_k)$ is defined as

$$\text{ncut}(P_1, \ldots, P_k) = \sum_{i=1}^{k} \frac{\text{out}(P_i)}{\text{within}(P_i) + \text{out}(P_i)}, \tag{1.2}$$

where

$$\text{within}(P_i) = \sum_{u,v \in P_i} w(u, v), \tag{1.3}$$

which measures the number of edges inside the subgraph induced by P_i (intra-connection); and

$$\text{out}(P_i) = \sum_{u \in P_i, v \in \mathcal{V} \setminus P_i} w(u, v) \tag{1.4}$$

measures the number of edges between P_i and the remaining nodes in the graph (inter-connection). Note that in the definition of $\mathrm{within}(P_i)$ (1.3), each edge within P_i is counted twice. In the special case $k = 2$, we have

$$\mathrm{out}(P_1) = \sum_{u \in P_1, v \in P_2} w(u, v) = \mathrm{out}(P_2) \stackrel{\text{def}}{=} \mathrm{cut}(P_1, P_2). \tag{1.5}$$

From Equation (1.2), it is evident that when each community has many more intra-connections than inter-connections, there is a small normalized cut.

Normalized cut (1.2) is a measurement of the extent that communities have more intra-connections than inter-connections and is shown to be an effective score [18]. One drawback of normalized cut is that it tends to increase when the number of communities increases. Specifically, we have the following theorem.

Theorem 1.1. *The normalized cut strictly increases when one community is split into two.*

Proof. Assume a community P is split into Q_1 and Q_2, then the associated increase of normalized cut is

$$\Delta_{ncut} = \frac{\mathrm{out}(Q_1)}{\mathrm{within}(Q_1) + \mathrm{out}(Q_1)} + \frac{\mathrm{out}(Q_2)}{\mathrm{within}(Q_2) + \mathrm{out}(Q_2)} - \frac{\mathrm{out}(P)}{\mathrm{within}(P) + \mathrm{out}(P)},$$

Denote $i_1 = \mathrm{within}(Q_1)$, $i_2 = \mathrm{within}(Q_2)$, $i_3 = \mathrm{within}(P)$, $o_1 = \mathrm{out}(Q_1)$, $o_2 = \mathrm{out}(Q_2)$, $o_3 = \mathrm{out}(P)$ and $c = \mathrm{cut}(Q_1, Q_2)$ and note that $o_3 = o_1 + o_2 - 2c$ and $i_3 = i_1 + i_2 + 2c$, then we have

$$\Delta_{ncut} = \frac{2c(i_1 + o_1)(i_2 + o_2) + o_2(i_1 + o_1)^2 + o_1(i_2 + o_2)^2}{(i_1 + o_1)(i_2 + o_2)(i_1 + i_2 + o_1 + o_2)} > 0.$$

Note that this proof does not say that more communities always corresponds to a larger normalized cut in general (i.e. when the communities are not obtained through recursive splitting). □

In practice, we observed that the normalized cut increases almost linearly with respect to the number of communities. Some community detection algorithms automatically determine the number of communities, hence it is not fair to compare normalized cut for such algorithms against others that detect a pre-assigned number of communities. Therefore, it makes more sense to use the average normalized cut, i.e. the normalized cut divided by the number of communities. In addition, since the average normalized cut can be treated as a per community property, it also applies to overlapping communities. Given k communities $P_1, \ldots, P_k$ (which may be overlapping), we define the average normalized cut as

$$\mathrm{AvgNcut}(P_1, \ldots, P_k) = \frac{1}{k} \sum_{i=1}^{k} \frac{\mathrm{out}(P_i)}{\mathrm{within}(P_i) + \mathrm{out}(P_i)}. \tag{1.6}$$

Conductance [19], which is shown to be an effective measure [18], is defined for a community as $\mathrm{Conductance}(P_i) = \frac{\mathrm{out}(P_i)}{\mathrm{within}(P_i)+\mathrm{out}(P_i)}$. Hence the average normalized cut is actually equal to the average conductance (per community).

External measures

True/False Positives/Negatives Assuming $\eta_i \in \{-1, 1\}$, $i = 1, \ldots, N$ are N unknown binary variables we would like to predict using certain algorithm, and the corresponding predicted values are $\xi_i \in \{-1, 1\}$, $i = 1, \ldots, N$. We call an observation of 1 a positive observation and an observation of -1 a negative observation. Then we can call a prediction ξ_i to be a true/false positive/negative as defined in Figure 1.1. By an abuse of notation, when we use TP, FP, TN, FN in a formula, they stand for the number of ξ_i's belonging to the corresponding category. Also, when we refer to the concept we will use roman font but in a formula we will use italic font. As we will see, many measures can be defined using these four concepts.

		True Value	
		$\eta_i = 1$	$\eta_i = -1$
Prediction	$\xi_i = 1$	True Positive (TP)	False Positive (FP)
	$\xi_i = -1$	False Negative (FN)	True Negative (TN)

Figure 1.1: Definition of true positive (TP), false positive (FP), true negative (TN) and false negative (FN).

Precision, Recall and F Scores The *precision*, *recall* and F *scores* are defined as

$$Precision = \frac{TP}{TP + FP}, \tag{1.7}$$

$$Recall = \frac{TP}{TP + FN}, \tag{1.8}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}. \tag{1.9}$$

Intuitively, precision measures the proportion of "positive" predictions that are correct; recall measures the proportion of "positive" true values that are correctly predicted; and F scores is a balance between the two. The most used F score is the F_1 score, which is the harmonic mean of precision and recall.

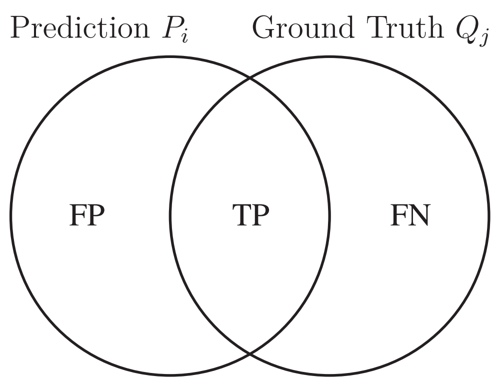
Defining Measures for Clustering For data set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, we assume the predicted clusters are $P_1, \ldots, P_k$ and the ground truth clusters are $Q_1, \ldots, Q_{k'}$. How can one define a measure for the quality of the predicted clusters? We start with the easiest case—comparing one cluster P_i from the predicted clusters to one cluster Q_j from the ground truth clusters. We define

$$\xi_l = \begin{cases} 1 & \boldsymbol{x}_l \in P_i \\ -1 & \text{otherwise} \end{cases} \quad \text{and} \quad \eta_l = \begin{cases} 1 & \boldsymbol{x}_l \in Q_j \\ -1 & \text{otherwise} \end{cases}$$

for $l = 1, \ldots, n$. With this definition of ξ_l and η_l, we can compute TP, FP, TN, FN as above. Specifically, $TP(P_i, Q_j) = |P_i \cap Q_j|$, $TP(P_i, Q_j) + FP(P_i, Q_j) = |P_i|$, $TP(P_i, Q_j) + FN(P_i, Q_j) = |Q_j|$, and therefore

$$
\begin{aligned}
\mathit{Precision}(P_i, Q_j) &= \mathit{Recall}(Q_j, P_i) = \frac{|P_i \cap Q_j|}{|P_i|}, && (1.10)\\
\mathit{Recall}(P_i, Q_j) &= \mathit{Precision}(Q_j, P_i) = \frac{|P_i \cap Q_j|}{|Q_j|}, && (1.11)\\
F_1(P_i, Q_j) &= F_1(Q_j, P_i) = \frac{2|P_i \cap Q_j|}{|P_i| + |Q_j|}, && (1.12)
\end{aligned}
$$

as illustrated in Figure 1.2. We can see that here precision measures how large the intersec-



$$
\begin{aligned}
\mathit{Precision}(P_i, Q_j) &= \frac{TP}{TP + FP} = \frac{|P_i \cap Q_j|}{|P_i|}\\
\mathit{Recall}(P_i, Q_j) &= \frac{TP}{TP + FN} = \frac{|P_i \cap Q_j|}{|Q_j|}\\
F_1(P_i, Q_j) &= \frac{2TP}{2TP + FP + FN} = \frac{2|P_i \cap Q_j|}{|P_i| + |Q_j|}
\end{aligned}
$$

Figure 1.2: Precision, recall and F_1 score for two clusters.

tion of P_i and Q_j is compared to P_i, while recall compares the intersection to Q_j and F_1 score compares it to both. Thus when F_1 score is close to 1, we can expect the predicted cluster is highly overlapped with the ground truth cluster. On the other hand, the precision can be 1 when P_i is fully contained in Q_j and the recall can be 1 when Q_j is fully contained in P_i.

Now we can define the average F_1 score for the whole clustering result as

$$
F_1 = \frac{1}{2}\left(\frac{1}{k}\sum_{i=1}^{k}\max_j F_1(P_i, Q_j) + \frac{1}{k'}\sum_{j=1}^{k'}\max_i F_1(Q_j, P_i)\right). \tag{1.13}
$$

The first term in the large parenthesis of (1.13) computes the F_1 score for each P_i with the best match, and takes the average of them. The second term swaps predicted clusters and ground truth clusters and does the same thing again. We can also define average precision and average recall in a similar fashion.

(Adjusted) Rand Index Rand index considers a different prediction problem—predicting whether a pair of data items belong to the same cluster. Assuming the same data set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, the same generated clusters $P_1, \ldots, P_k$ and the same ground truth clusters $Q_1, \ldots, Q_{k'}$ as in previous paragraphs, unlike for F scores, we define ξ_{lr}'s and η_{lr}'s as

$$\xi_{lr} = \begin{cases} 1 & \exists i \text{ s.t. } \boldsymbol{x}_l \in P_i \text{ and } \boldsymbol{x}_r \in P_i \\ -1 & \text{otherwise} \end{cases}$$

and

$$\eta_{lr} = \begin{cases} 1 & \exists j \text{ s.t. } \boldsymbol{x}_l \in Q_j \text{ and } \boldsymbol{x}_r \in Q_j \\ -1 & \text{otherwise} \end{cases}$$

for all $1 \leq l \neq r \leq n$. Then TP, FP, TN, FN can be computed using these ξ_{lr}'s and η_{lr}'s, and they have different meanings as illustrated in Table 1.1, than the ones in F scores.

Table 1.1: Meaning of TP, FP, TN, FN in the definition of Rand index.

Given any two data items, do they belong to the same cluster...

in the ground truth?	in the computed clusters?	result
Yes	Yes	TP
No	No	TN
Yes	No	FN
No	Yes	FP

Rand index is then defined as

$$RI = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n(n-1)/2}, \tag{1.14}$$

which computes the proportion of true predictions among all the predictions of whether a pair of data items belongs to the same cluster.

Adjusted Rand index is a popular variant of Rand index, which tries to avoid "spuriously large obtained values of the index" by correcting for "chance levels of agreement" [20]. Conceptually, the adjusted Rand index is formed as

$$ARI = \frac{RI - \mathbb{E}(RI)}{\max RI - \mathbb{E}(RI)} = \frac{RI - \mathbb{E}(RI)}{1 - \mathbb{E}(RI)}. \tag{1.15}$$

Under certain probabilistic assumptions [21], ARI can be computed as [20]

$$ARI = \frac{\binom{n}{2}(TP + TN) - [(TP + FP)(TP + FN) + (TN + FP)(TN + FN)]}{\binom{n}{2}^2 - [(TP + FP)(TP + FN) + (TN + FP)(TN + FN)]}. \tag{1.16}$$

One should note that the definition of precision, recall, F_1 score and Rand index also apply to the case where the clusters have overlapping. However, the derivation of $\mathbb{E}(RI)$ requires the assumption that the clusters are disjoint (non-overlapping) [21]. Therefore, the ARI formula (1.16) cannot be applied to overlapping clusters. The formula of $\mathbb{E}(RI)$ for overlapping case is not known yet. For overlapping case, Collins and Dent [20] proposed a new measure called Omega index, inspired by adjusted Rand index. For each pair of data items, they count the number of clusters in which the pair is together, and a pair make a positive contribution to the Omega index if and only if this number is the same in generated and ground truth clusters. Therefore, if a pair is contained in 6 clusters in the generated result and 7 clusters in the ground truth, such pair would not help increasing the Omega index. In this dissertation, we consider such partial recovery of overlapping clusters still valuable and thus Omega index will not be used in this dissertation.

(Normalized) Mutual Information Mutual information comes from a information theory background. It measures the mutual dependence between two random variables. More specifically, it measures "the reduction in the uncertainty of one random variable due to the knowledge of the other" [22]. Let W and Z be two discrete random variables, then the mutual information of W and Z is defined as

$$MI(W,Z) = \sum_{w,z} p(w,z) \log\left(\frac{p(w,z)}{p(w)p(z)}\right), \tag{1.17}$$

where $p(w,z)$ is the joint probability function of W and Z, and $p(w)$ and $p(z)$ are the marginal probability distribution functions of W and Z, respectively. Now we assume again the data set to be $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, the generated clusters to be $P_1, \ldots, P_k$ and the ground truth clusters to be $Q_1, \ldots, Q_{k'}$.

We first assume $P_1, \ldots, P_k$ and $Q_1, \ldots, Q_{k'}$ to be two *partitions* of the data set X, which means $P_i \cap P_j = \emptyset,\ \forall i \neq j$ and $\bigcup_i P_i = X$ and the same for Q_j's. Under this assumption we define random variable W and Z with joint distribution

$$\mathbb{P}(W=i, Z=j) = \frac{|P_i \cap Q_j|}{n}, \quad i = 1, \ldots, k,\ j = 1, \ldots, k', \tag{1.18}$$

and the marginal distribution can be computed accordingly as

$$\mathbb{P}(W=i) = \frac{|P_i|}{n}, \quad i = 1, \ldots, k, \tag{1.19}$$

$$\mathbb{P}(Z=j) = \frac{|Q_j|}{n}, \quad j = 1, \ldots, k'. \tag{1.20}$$

Intuitively, if we randomly pick a data item $\boldsymbol{x_l}$ from X, there will exist unique $W \in \{1, \ldots, k\}$ and $Z \in \{1, \ldots, k'\}$ such that $\boldsymbol{x_l} \in P_W$ and $\boldsymbol{x_l} \in Q_Z$. Then such W and Z will follow the distribution defined in (1.18). With such defined W and Z, knowing P_W will give us $MI(W,Z)$ bits of information of Q_Z.

In practice the normalized mutual information (NMI) are usually used to "facilitate interpretation and comparison across different conditions" and "improve the sensitiveness of MI with respect to the difference in cluster distribution in the two clusterings" [23]. In 1987, several normalized variants of mutual information (NMI) [24] was proposed in pure information theoretic sense. In 2000, Strehl et al. [25] proposed to use MI for (non-overlapping) clustering comparing, and later Strehl and Ghosh [26, 27] proposed to use NMI. The most seen version of NMI in recent literatures, which was proposed by [28], is defined as

$$NMI(W, Z) = \frac{2MI(W, Z)}{H(W) + H(Z)}, \tag{1.21}$$

where $H(W)$ is the entropy of W and is defined as

$$H(W) = -\sum_{w} p(w) \log p(w) = -\sum_{i=1}^{k} \mathbb{P}(W = i) \log \mathbb{P}(W = i), \tag{1.22}$$

and $H(Z)$ can be computed similarly.

Now we come to the case where $P_1, \ldots, P_k$ and $Q_1, \ldots, Q_{k'}$ are overlapping clusters. Specifically, $P_i \cap P_j$ might not be $\emptyset$ for some $i \neq j$ and $\bigcup_i P_i$ might not be X (same for Q_j's'). Due to sum-to-one constraint of probability distributions, the above formulation of NMI does not work for overlapping clusters. Lancichinetti et al. [29] proposed a version of NMI that can be computed for overlapping clusters in the following way.

1. Define a series of Bernoulli random variables: $W_1, \ldots, W_k$ and $Z_1, \ldots, Z_{k'}$ with the following joint distribution.

$$\mathbb{P}(W_i = 1, Z_j = 1) = \frac{|P_i \cap Q_j|}{n}, \tag{1.23}$$

$$\mathbb{P}(W_i = 1, Z_j = 0) = \frac{|P_i| - |P_i \cap Q_j|}{n}, \tag{1.24}$$

$$\mathbb{P}(W_i = 0, Z_j = 1) = \frac{|Q_j| - |P_i \cap Q_j|}{n}, \tag{1.25}$$

$$\mathbb{P}(W_i = 0, Z_j = 0) = \frac{n - |P_i \cup Q_j|}{n}. \tag{1.26}$$

2. For each $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, k'\}$, compute conditional entropy $H(W_i|Z_j) = \sum_{w_i, z_j} = p(w_i, z_j) \log \frac{p(z_j)}{p(w_i, z_j)}$.

3. For each i, compute $\widetilde{H}(W_i|\boldsymbol{Z})$ by

$$\widetilde{H}(W_i|\boldsymbol{Z}) = \begin{cases} H(W_i) & J(i) = \emptyset \\ \min_{j \in J(i)} H(W_i|Z_j) & \text{otherwise}, \end{cases} \tag{1.27}$$

where $J(i) = \{j \in \{1, \ldots, k'\} : h[P(1,1)] + h[P(0,0)] > h[P(0,1)] + h[P(1,0)]\}$, $P(w,z) = P(W_i = w, Z_j = z)$, and $h(p) = -p \log p$.

4. Compute $\widetilde{H}(\boldsymbol{W}|\boldsymbol{Z})_{\text{norm}}$ by

$$\widetilde{H}(\boldsymbol{W}|\boldsymbol{Z})_{\text{norm}} = \frac{1}{k} \sum_{i=1}^{k} \frac{H(W_i|\boldsymbol{Z})}{H(W_i)}. \tag{1.28}$$

5. Repeat the above steps to compute $\widetilde{H}(\boldsymbol{Z}|\boldsymbol{W})_{\text{norm}}$

6. Finally compute overlapping NMI as

$$\widetilde{NMI} = 1 - \frac{1}{2}[\widetilde{H}(\boldsymbol{W}|\boldsymbol{Z})_{\text{norm}} + \widetilde{H}(\boldsymbol{Z}|\boldsymbol{W})_{\text{norm}}]. \tag{1.29}$$

Here, we add tilde to H and NMI when they do not have the actual information theoretical meaning. We notice that the $\widetilde{NMI}$ defined here shares some similar steps as previously defined average F_1 score: defining pairwise scores, finding best match, averaging scores of best match, switching computed and ground truth clusters and taking average again. However, unlike F_1 score, the pairwise score here ($H(W_i|Z_j)$) is more mysterious in how it measures the difference/similarity between two clusters. For example, we consider the following two cases, both with $n = 100$

1. $|P_i| = 1,\ |Q_j| = 1,\ |P_i \cap Q_j| = 0$

2. $|P_i| = 99,\ |Q_j| = 99,\ |P_i \cap Q_j| = 98$

In both cases, we have $H(W_i|Z_j) \approx 0.0806$ and $H(W_i|Z_j)/H(W_i) \approx 0.9982$. Therefore, they make the same contribution in the $\widetilde{NMI}$. However, Case 1 are two totally different clusters but Case 2 are two almost same clusters. Due to such mysterious behaviors, we do not use this overlapping version of NMI in this dissertation.

1.3 Contributions and Outlines

The key contributions of this dissertation are:

- Development of DC-NMF method [30], which improves the per iteration time complexity of standard NMF [31] from $O(kN + t(k^3 + (m+n)k^2))$ to $O(k(N+m+n))$, where $m \times n$ is the size of the input matrix, N is the number of non-zeros of the input matrix, k is the reduced rank, and t is the number of steps for finding the optimal active set.

- Development of HierSymNMF2 [32] method which improves the per iteration time complexity of symmetric NMF [11] from $O(kN + t(k^3 + 2nk^2))$ to $O(k(N + 2n)$, where $n \times n$ is the size of the input matrix, N is the number of non-zeros of the input matrix, k is the reduced rank, and t is the number of steps for finding the optimal active set.

- Design and implementation of novel JointNMF algorithm [33] that is able to analyze hybrid data that has both content and connection information.

- Novel application of JointNMF on text based link prediction and leader/activity detection in organizations.

- Construction of new DBLP data sets with ground truth communities and new hybrid data sets from US patents with ground truth clusters.

Note that DC-NMF, HierSymNMF2 and HierNMF2 [10] all shares the same divisive clustering framework, where one recursively choose a cluster and split it into two new clusters $(k-1)$ times. The distinct features for each divisive clustering algorithm are how one cluster is split into two and at each iteration how to choose a cluster for next split (i.e. the splitting criterion). HierSymNMF2 is different from the other two because it uses rank-2 SymNMF instead of NMF for splitting a cluster and it has distinct graph-based splitting criterion. Both HierNMF2 and DC-NMF are accelerated version of standard NMF, but they have the following major differences:

- After finding k clusters, HierNMF2 will stop but DC-NMF will further collect representative vectors from each leaf node to form a rank-k W and then compute H to obtain a solution of rank-k NMF.
- The splitting criterion proposed in [10] for HierNMF2 is specifically designed for document clustering, while the splitting criterion used in this thesis considers low rank approximation and leads to provable error bound.
- HierNMF2 deletes outliers to improve clustering quality, while DC-NMF keeps all the data and features to obtain a complete factorization of the input matrix.

An outline and detailed contributions of each chapter are given below.

In Chapter 2, we summarize the fundamental numerical routines for NMF and its variants that are used throughout this dissertation: BPP method for general nonnegative least squares (NNLS) [31] and its improved version for rank-2 NNLS [10].

In Chapter 3, we develop a fast algorithm for computing NMF using a divide-and-conquer strategy, called DC-NMF. Given an input matrix where the columns represent data items, we build a binary tree structure of the data items using an efficient algorithm for computing rank-2 NMF (Chapter 2), and then gather information from the tree to initialize the rank-k NMF, which needs only a few iterations to reach a desired solution. We also investigate various criteria for selecting the node to split when growing the tree. We demonstrate the scalability of our algorithm for computing general rank-k NMF as well as its effectiveness in clustering and topic modeling for large-scale text data sets, by comparing it to other frequently utilized state-of-the-art algorithms. The value of the proposed approach lies in the highly efficient and accurate method for initializing rank-k NMF and the scalability achieved from the divide-and-conquer approach of the algorithm and properties of rank-2 NMF. In summary, we present efficient tools for analyzing large-scale data sets, and techniques that can be generalized to many other data analytics problem domains. The content of this chapter is published in [30].

In Chapter 4, we apply a divide-and-conquer strategy to discover hierarchical community structure, non-overlapping within each level. The algorithm is based on the highly efficient Rank-2 Symmetric Nonnegative Matrix Factorization. Empirical results have shown that our algorithm has competitive overall efficiency, leading performance in minimizing the average normalized cut, and that the non-overlapping communities found by our algorithm recover the ground-truth communities better than state-of-the-art algorithms for overlapping community detection. In addition, we present a new data set of the DBLP computer science bibliography network with richer meta-data and verifiable ground-truth knowledge, which can foster future research in community finding and interpretation of communities in large networks. The content of this chapter is published in [32].

In Chapter 5, we present a hybrid method called JointNMF which is applied to latent information discovery from data sets that contain both text content and connection structure information. The new method jointly optimizes an integrated objective function, which is a combination of two components: the Nonnegative Matrix Factorization (NMF) objective function for handling text content and the Symmetric NMF (SymNMF) objective function for handling network structure information. An effective algorithm for the joint NMF objective function is proposed so that the efficient method of block coordinate descent (BCD) framework can be utilized. The proposed hybrid method simultaneously discovers content associations and related latent connections without any need for postprocessing of additional clustering. It is shown that the proposed method can also be applied when the text content is associated with hypergraph edges. An additional capability of the JointNMF is prediction of unknown network information which is illustrated using several real world problems such as citation recommendations of papers and leader/activity detection in organizations. The proposed method can also be applied to general data expressed with both feature space vectors and pairwise similarities. The content of this chapter is published in [33].

In Chapter 6, we briefly study the relation between NMF, SymNMF and JointNMF. Finally in Chapter 7 we conclude the whole dissertation.

CHAPTER 2

FUNDAMENTAL NUMERICAL ROUTINE: ACTIVE-SET-TYPE ALGORITHMS FOR NONNEGATIVE LEAST SQUARES

In this dissertation, all variants of NMF are solved using a block coordinate descent (BCD) framework [34]. BCD is an algorithm framework for non-linear optimization problem

$$\min_{\boldsymbol{y}\in\mathcal{Y}} f(\boldsymbol{y}), \tag{2.1}$$

where $\mathcal{Y}$ is a closed convex subset of $\mathbb{R}^N$. A BCD algorithm utilizes a proper Cartesian decomposition of $\mathcal{Y}$:

$$\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_q, \tag{2.2}$$

where each $\mathcal{Y}_i$, called a block, is a closed convex subset of $\mathbb{R}^{N_i}$ satisfying $N = \sum_{i=1}^{q} N_i$. The BCD method, as its name, is a block version of coordinate descent method. Precisely, assuming $\boldsymbol{y}^{(j)} = (\boldsymbol{y}_1^{(j)}, \ldots, \boldsymbol{y}_q^{(j)})$ is the current iterate at the jth step, the BCD method computes the next iterate $\boldsymbol{y}^{(j+1)} = (\boldsymbol{y}_1^{(j+1)}, \ldots, \boldsymbol{y}_q^{(j+1)})$ by solving the following subproblem block by block:

$$\boldsymbol{y}_i^{(j+1)} \leftarrow \arg\min_{\boldsymbol{\eta}\in\mathcal{Y}_i} f\left(\boldsymbol{y}_1^{(j+1)}, \ldots, \boldsymbol{y}_{i-1}^{(j+1)}, \boldsymbol{\eta}, \boldsymbol{y}_{i+1}^{(j)}, \ldots \boldsymbol{y}_q^{(j)}\right). \tag{2.3}$$

With carefully chosen blocks, BCD algorithms can have very good convergence property, as stated in the following theorem.

Theorem 2.1 ([34, 35, 36])**.** *Suppose f is continuously differentiable in $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_q$, where each $\mathcal{Y}_i$, $i = 1, \ldots, q$, are closed convex sets. Furthermore, suppose that for all i and j, the minimum of* (2.3) *is uniquely attained. Let $\{\boldsymbol{y}^{(j)}\}$ be the sequence generated by the BCD method in* (2.3)*. Then, every limit point of $\{\boldsymbol{y}^{(j)}\}$ is a stationary point. The uniqueness* *of the minimum is not required when q is two.*

For example, a good BCD algorithm to solve NMF formulation (1.1) is the alternating nonnegative least squares (ANLS) scheme, which solves the following two subproblem alternatively until convergence:

$$\min_{W \geq 0} \|H^{\intercal} W^{\intercal} - X^{\intercal}\|_F, \tag{2.4}$$

$$\min_{H \geq 0} \|WH - X\|_F. \tag{2.5}$$

As we can see in (2.4), (2.5) and other NMF algorithms in later chapters, a fundamental problem that is solved in every iteration of many BCD algorithms for NMF is the following nonnegative least square (NNLS) problem:

$$\min_{Y \geq 0} \|FY - G\|_F, \tag{2.6}$$

where $F \in \mathbb{R}_+^{m \times k}$, $Y \in \mathbb{R}_+^{k \times n}$, and $G \in \mathbb{R}_+^{m \times n}$.

This dissertation uses a state-of-the-art algorithm called block principal pivoting (BPP) method [31] to solve the NNLS problem, as restated in Algorithm 1. The time complexity of Algorithm 1 is upper bounded by [10]:

$$4kN + 2(m+n)k^2 + t[(1/3)k^3 + 2(m+n)k^2] \text{ flops}, \tag{2.7}$$

where N is the number of nonzeros in G and t is the number of iterations (the while loop) for searching the optimal active set. We can see that the time complexity is linear with respect to m and n, which is good but it is also cubic with respect to k. In some applications where k is large, this algorithm will become slow. For NMF clustering algorithms, one important strategy is divide-and-conquer, utilizing that binary clustering/splitting using NNLS for $k = 2$ is always fast. To find k clusters, one just need $k - 1$ recursive binary splits, which

Algorithm 1 BPP method for NNLS problem (2.6). $Y(\mathcal{A}_j, j)$ and $Z(\mathcal{B}_j, j)$ represents the subsets of j-th column of Y and Z indexed by $\mathcal{A}_j$ and $\mathcal{B}_j$, respectively.

Input: $F \in \mathbb{R}^{m\times k}$, $G \in \mathbb{R}^{m\times n}$
Output: $Y = \arg\min_{Y\geq 0} \|FY - G\|_F \in \mathbb{R}^{k\times n}$

Compute $F^\intercal F$ and $F^\intercal G$.
Initialize $\mathcal{A}_j \leftarrow \emptyset$ and $\mathcal{B}_j \leftarrow \{1, \dots, k\}$ for all $j \in \{1, \dots, n\}$. Set $Y \leftarrow 0$, $Z \leftarrow -F^\intercal G$, $\alpha(\in \mathbb{R}^n) \leftarrow 3$, and $\beta(\in \mathbb{R}^n) \leftarrow k + 1$.
Compute $Y(\mathcal{A}_j, j)$ and $Z(\mathcal{B}_j, j)$ for all $j \in \{1, \dots, n\}$ using the following formula with column grouping:

$$Y(\mathcal{A}_j, j) \leftarrow \operatorname*{arg\,min}_{\boldsymbol{\eta}\in\mathbb{R}^{|\mathcal{A}_j|}} \|F(:, \mathcal{A}_j)\boldsymbol{\eta} - G(:, j)\|_2,$$
$$Z(\mathcal{B}_j, j) \leftarrow F(:, \mathcal{B}_j)^\intercal(F(:, \mathcal{A}_j)\boldsymbol{\eta} - G(:, j)).$$

while $Y(\mathcal{A}_j, j) \geq 0$ or $Z(\mathcal{B}_j, j) \geq 0$ are not satisfied for any j **do**
  Let $\mathcal{I} \leftarrow \{j : Y(\mathcal{A}_j, j) \geq 0$ or $Z(\mathcal{B}_j, j) \geq 0$ are not satisfied$\}$
  Compute $\mathcal{C}_j$ for all $j \in \mathcal{I}$ by

$$\mathcal{C}_j \leftarrow \{i \in \mathcal{A}_j : Y(i, j) < 0\} \cup \{i \in \mathcal{B}_j : Z(i, j) < 0\}.$$

  For all $j \in \mathcal{I}$ with $|\mathcal{C}_j| < \beta_j$, set $\beta_j \leftarrow |\mathcal{C}_j|$, $\alpha_j \leftarrow 3$ and $\hat{\mathcal{C}}_j \leftarrow \mathcal{C}_j$.
  For all $j \in \mathcal{I}$ with $|\mathcal{C}_j| \geq \beta_j$ and $\alpha_j \geq 1$, set $\alpha_j \leftarrow \alpha_j - 1$ and $\hat{\mathcal{C}}_j \leftarrow \mathcal{C}_j$
  For all $j \in \mathcal{I}$ with $|\mathcal{C}_j| \geq \beta_j$ and $\alpha_j = 0$, set $\hat{\mathcal{C}}_j \leftarrow \{\max(\mathcal{C}_j)\}$.
  Update $\mathcal{A}_j$ and $\mathcal{B}_j$ for all $j \in \mathcal{I}$ by

$$\mathcal{A}_j \leftarrow (\mathcal{A}_j - \hat{\mathcal{C}}_j) \cup (\hat{\mathcal{C}}_j \cap \mathcal{B}_j),$$
$$\mathcal{B}_j \leftarrow (\mathcal{B}_j - \hat{\mathcal{C}}_j) \cup (\hat{\mathcal{C}}_j \cap \mathcal{A}_j).$$

  Update $Y(\mathcal{A}_j, j)$ and $Z(\mathcal{B}_j, j)$ for $j \in \mathcal{I}$ using the following formula with column grouping:

$$Y(\mathcal{A}_j, j) \leftarrow \operatorname*{arg\,min}_{\boldsymbol{\eta}\in\mathbb{R}^{|\mathcal{A}_j|}} \|F(:, \mathcal{A}_j)\boldsymbol{\eta} - G(:, j)\|_2,$$
$$Z(\mathcal{B}_j, j) \leftarrow F(:, \mathcal{B}_j)^\intercal(F(:, \mathcal{A}_j)\boldsymbol{\eta} - G(:, j)).$$

end while

have the time complexity upper bounded by:

$$(k-1)[8N + 8(m+n) + t(8/3 + 8m + 8n)] \text{ flops}, \tag{2.8}$$

which is linear with respect to k. The BPP method with $k = 2$ can be further accelerated by reducing the overhead of searching the optimal active set, utilizing special properties of NNLS when $k = 2$.

When $k = 2$, the objective function of single right hand NNLS problem is $J(\boldsymbol{y}) \stackrel{\text{def}}{=} \|F\boldsymbol{y} - \boldsymbol{g}\|_2^2 = \|\boldsymbol{f}_1 y_1 + \boldsymbol{f}_2 y_2 - \boldsymbol{g}\|_2^2$, where $F = [\boldsymbol{f}_1, \boldsymbol{f}_2] \in \mathbb{R}_+^{m\times 2}$, $\boldsymbol{g} \in \mathbb{R}_+^{m\times 1}$, and $\boldsymbol{y} = [y_1, y_2]^\intercal \in \mathbb{R}^{2\times 1}$, and the number of possible active sets is reduced to $2^2 = 4$. Unlike in a standard iterative optimization algorithm such as the projected gradient descent (PGD) method where the algorithm strucure is not directly affected by the value of k, in an active-set method, when $k = 2$, we can directly and effectively obtain the optimal active set by choosing the one with the smallest $J(\boldsymbol{y})$ among all the feasible solutions $\boldsymbol{y} \geq 0$ as follows. If the solution $\boldsymbol{y}^{\emptyset} = \arg\min_{\boldsymbol{y}} \|F\boldsymbol{y} - \boldsymbol{g}\|_2$ for the unconstrained problem is nonnegative, then $\boldsymbol{y}^{\emptyset}$ is the solution for the nonnegativity constrained problem. Otherwise, between the solutions for the two unconstrained problems $\min \|\boldsymbol{f}_i y_i - \boldsymbol{g}\|_2$ $(i = 1, 2)$, which are always feasible since $\boldsymbol{f}_i \geq 0$ and $\boldsymbol{g} \geq 0$, we can efficiently choose the best one. We exclude $\boldsymbol{y} = (0, 0)^\intercal$ since one of the above three is always better. For NLS with multiple right-hand sides, it is not cache-efficient to compute the solutions for the above three cases separately. Better computational efficiency emerges in our algorithm when we solve NLS with n right hand side vectors $\mathbf{y}_i$ simultaneously, which is summarized in Algorithm 2. The entire for-loop (lines 5-15, Algorithm 2) is embarrassingly parallel and can be vectorized. To achieve this, unconstrained solutions for all three possible passive sets are computed before entering the for-loop. Algorithm 2 represents a non-random pattern of memory access, and is much faster for Rank-2 NMF than applying existing active-set-type algorithms directly. It further improves the upper bound (2.8) to

$$(k-1)\cdot[8N+8(m+n)+6n] \text{ flops} \tag{2.9}$$

In Chapter 3 and Chapter 4, we will explore the details of the divide-and-conquer strategy.

Algorithm 2 Fast algorithm for solving (2.6) with $k=2$, where $F=[\boldsymbol{f}_1,\boldsymbol{f}_2]\in\mathbb{R}^{m\times 2}$

Input: $F\in\mathbb{R}^{m\times 2}$, $G\in\mathbb{R}^{m\times n}$
Output: $Y=\arg\min_{Y\geq 0}\|FY-G\|_F\in\mathbb{R}^{2\times n}$

Solve unconstrained least squares $Y^{\emptyset}=[\boldsymbol{y}_1^{\emptyset},\ldots,\boldsymbol{y}_n^{\emptyset}]\leftarrow\min\|FY-G\|_F^2$ by normal equation $F^{\intercal}FY=F^{\intercal}G$.
$\beta_1\leftarrow\|\boldsymbol{f}_1\|_2$, $\beta_2\leftarrow\|\boldsymbol{f}_2\|_2$
$\boldsymbol{u}\leftarrow(G^{\intercal}\boldsymbol{f}_1)/\beta_1^2$
$\boldsymbol{v}\leftarrow(G^{\intercal}\boldsymbol{f}_2)/\beta_2^2$
for $i=1$ to n **do**
    if $\boldsymbol{y}_i^{\emptyset}\geq 0$ **then**
        $Y(:,i)\leftarrow\mathbf{g}^{\emptyset}$
    else
        if $u_i\beta_1\geq v_i\beta_2$ **then**
            $Y(:,i)\leftarrow[u_i,0]^{\intercal}$
        else
            $Y(:,i)\leftarrow[0,v_i]^{\intercal}$
        end if
    end if
end for

CHAPTER 3

DC-NMF FOR LARGE SCALE FEATURE-BASED CLUSTERING

As mentioned in Chapter 2, the time complexity of ANLS-BPP method is cubic with respect to k, and a solution to such slow down is to apply a divide-and-conquer strategy. In this chapter we will discuss this idea in detail and develop it into a fast algorithm for NMF called DC-NMF. In Section 3.1, we will study some properties of rank-2 NMF and justify its usage in DC-NMF. In Section 3.2 we fully describe the DC-NMF algorithm. And finally in Section 3.3 we demonstrate the effectiveness of DC-NMF via extensive experiments.

3.1 Rank-2 Approximation by SVD and NMF

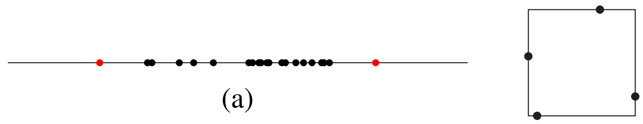
DC-NMF relies on certain properties of NMF when $k = 2$, which makes applying divide-and-conquer possible. In Chapter 2, we have seen some algorithmic benefits of rank-2 NMF and divide-and-conquer strategy. In this section, we offer some theoretical justification for rank-2 NMF.

Due to the additional constraints, the approximation error of NMF is lower bounded by

$$\min_{W \geq 0, H \geq 0} \|X - WH\|_F \geq \min_{W,H} \|X - WH\|_F. \tag{3.1}$$

By Eckart-Young Theorem [37], the right hand side is equal to the approximation error of SVD. When $k = 1$, NMF is the same as SVD, due to the well known Perron-Frobenius Theorem [38]. It follows from the theorem that there are nonnegative left and right singular vectors associated with the leading singular value of $X \in \mathbb{R}_+^{m \times n}$.

When $k > 1$, the low rank approximations by NMF and SVD are not the same in general. We define the rank and nonnegative rank of a matrix using a single framework: For a $X \in \mathbb{R}^{m \times n}$, $\mathrm{rank}(X)$ is the smallest integer p for which there exist $U \in \mathbb{R}^{m \times p}$ and



(a) (b)

Figure 3.1: Illustration of NMF of rank-2 and rank-3 matrices. (a) When a nonnegative matrix has rank 2, its columns can be projected onto a one-dimensional subspace, and we can find two extreme endpoints (the red dots in this figure) which define a convex hull that encloses all the points. (b) The columns of a matrix $X \in \mathbb{R}_+^{4\times 4}$ with rank 3 (four solid dots in the figure) are projected onto a two-dimensional subspace. The depicted region is the intersection of the 3-dimensional simplex with this two-dimensional subspace, with four sides on the boundary. There are no three vertexes that define a convex hull that encloses all the four points, and thus $\text{rank}_+(X) > 3$.

$V \in \mathbb{R}^{p\times n}$ such that $X = UV$. The nonnegative rank, $\text{rank}_+(X)$, is the smallest integer p for which there exist $U \in \mathbb{R}_+{}^{m\times p}$ and $V \in \mathbb{R}_+{}^{p\times n}$ such that $X = UV$. The nonnegative rank of a matrix $X \in \mathbb{R}_+^{m\times n}$ is the same as the smallest possible number of vertices of a convex hull that contain all columns of X when projected onto the $(m-1)$-dimensional simplex [39]. This relationship has an important implication for rank-2 NMF, as illustrated below. Cohen and Rothblum [40] showed an interesting relationship between SVD and NMF: given $X \in \mathbb{R}_+^{m\times n}$, if $\text{rank}(X) = 2$, then $\text{rank}_+(X) = 2$. They also provided a constructive method to generate rank-2 NMF when $\text{rank}(X) = 2$. Without loss of generality, we can assume that $\|X(:,i)\|_1 = 1$ for every column of X. Under this assumption, when $k = 2$, all columns of X lie on a one-dimensional simplex; therefore, there must exist two columns of X that define two extreme rays of a 2-d nonnegative cone which encloses all the columns (Figure 3.1(a)). When $\text{rank}(X) > 2$, this property does not hold in general. For example, when $\text{rank}(X) = 3$, under the sum-to-one constraint, the columns of X lie on a two-dimensional subspace, and there is not always a convex hull with three vertices that encloses all columns of X (Figure 3.1(b)).

Although the solution of rank-2 NMF can be computed based on the constructive proof for a theorem provided in [40], its usage is limited to the case with a nonnegative matrix X with $\text{rank}(X) = 2$. When $\text{rank}(X) > 2$, one may consider computing its rank-2 approximation by SVD $\hat{X} = U_2 \Sigma_{2\times 2} V_2^{\intercal}$ first. One difficulty is that $\hat{A}$ will not necessarily be nonnegative although $\mathrm{rank}(\hat{X}) = 2$. On the other hand, simply setting the negative elements in $\hat{X}$ to zero will change its rank.

We have observed empirically that when the reduced rank $k = 2$, the relative difference between the approximation errors by SVD and NMF were very small. The difference was measured by $|(error_{\mathrm{nmf}} - error_{\mathrm{svd}})/error_{\mathrm{svd}}|$ (Table 3.1), where $error_{\mathrm{nmf}}$ and $error_{\mathrm{svd}}$ are the approximation errors by NMF and SVD, respectively. We also noticed that this relative difference becomes larger as k increases.

Table 3.1: Relative difference in the approximation errors produced by SVD and NMF for sparse matrices of various sizes with 1% non-zero entries uniformly distributed in the interval $(0, 1)$. For each pair of (m, n), the results were the average over 100 random matrices $X \in \mathbb{R}_{+}^{m\times n}$. The approximation error of NMF was computed by taking the minimum of 20 random runs for each matrix.

	n \ m	300	500	1000	3000
$k = 2$	250	0.7704×10^{-4}	1.3296×10^{-4}	1.8039×10^{-4}	1.6177×10^{-4}
	300	1.0103×10^{-4}	1.4303×10^{-4}	1.7583×10^{-4}	1.6051×10^{-4}

	n \ m	300	500	1000	3000
$k = 3$	250	2.0238×10^{-4}	2.9706×10^{-4}	3.5395×10^{-4}	3.4395×10^{-4}
	300	2.4668×10^{-4}	3.0484×10^{-4}	3.6593×10^{-4}	3.3824×10^{-4}

3.2 Fast NMF based on Divide-and-Counquer

In this section, we propose a fast algorithm for computing NMF for any given $k \geq 2$, which we call DC-NMF (Divide-and-Conquer NMF). Based on the fast rank-2 NMF algorithm and a divide-and-conquer method, DC-NMF computes a high quality W for NMF, which we show in the following subsection. We will also provide and compare several alternative formulations for DC-NMF.

The value of k represents the number of clusters or number of topics, which is often larger than 2. In addition, since a larger k value produces a better low rank approximation, a fast algorithm that works for $k > 2$ is needed. Increasing the reduced rank k in the unconstrained low rank approximation (SVD) strictly improves the approximation quality until k reaches rank(X) [41]. For NMF, the following similar result holds.

Theorem 3.1. *For $X \in \mathbb{R}_+^{m\times n}$ with $\operatorname{rank}_+(X) = k$, we have*

$$\min_{W^{(p+1)}\geq 0, H^{(p+1)}\geq 0} \|X - W^{(p+1)}H^{(p+1)}\|_F < \min_{W^{(p)}\geq 0, H^{(p)}\geq 0} \|X - W^{(p)}H^{(p)}\|_F,$$

for all $p < k$, where $W^{(p)} \in \mathbb{R}_+^{m\times p}$ and $H^{(p)} \in \mathbb{R}_+^{p\times n}$.

Proof. Let $(W_*^{(p)}, H_*^{(p)}) = \arg\min_{W^{(p)}\geq 0, H^{(p)}\geq 0} \|X - W^{(p)}H^{(p)}\|_F$, and $R^{(p)} = (r_{ij})_{m\times n} = X - W_*^{(p)}H_*^{(p)}$. For $p < k$, we have $R^{(p)} \neq 0$, and we can prove at least one element of $R^{(p)}$ is positive. Assume $r_{ij} > 0$ for some i, j. Then we have

$$\|R^{(p+1)}\|_F \leq \|X - \begin{bmatrix} W_*^{(p)} & r_{ij}\boldsymbol{e}_i^m \end{bmatrix} \begin{bmatrix} H_*^{(p)} \\ (\boldsymbol{e}_j^n)^\intercal \end{bmatrix} \|_F < \|R^{(p)}\|_F,$$

where e_i^m (e_i^n) is the i-th unit vector in $\mathbb{R}^m$ ($\mathbb{R}^n$).

Now we prove that $R^{(p)}$ has at least one positive element. Assume $R^{(p)} \neq 0$ and has no positive element. Then any nonzero column, say jth column, of $R^{(p)}$: $R_{:j}^{(p)} = (r_{1j}, \ldots, r_{mj})^\intercal \neq 0$ has at least one negative element. Let's choose the greatest negative component, say $r_{ij} < 0$. Since $a_{i,j} \geq 0$, $r_{ij} < 0$, and

$$R_{:j}^{(p)} = [r_{1j}, \ldots, r_{mj}]^\intercal = [x_{1j}, \ldots, x_{mj}]^\intercal - \sum_{l=1}^{p} [w_{1l}, \ldots, w_{ml}]^\intercal h_{lj} \leq 0,$$

there always exists an index $\hat{l}$, such that $h_{\hat{l}j} \neq 0$ and $w_{i\hat{l}} \neq 0$. Then we can choose a small enough $\epsilon > 0$ and replace $h_{\hat{l}j}$ by $\tilde{h}_{\hat{l}j} = h_{\hat{l}j} - \epsilon > 0$ such that

$$\|[\tilde{r}_{1j}, \ldots, \tilde{r}_{mj}]^\intercal\|_F = \|[r_{1j}, \ldots, r_{mj}]^\intercal + \epsilon[w_{1\hat{l}}, \ldots, w_{m\hat{l}}]^\intercal\|_F < \|[r_{1j}, \ldots, r_{mj}]^\intercal\|_F .$$

However, this contradicts the assumption that $R^{(p)}$ is minimized. □

The above theorem shows that in the context of NMF for a nonnegative matrix, the approximation error is strictly reduced when the reduced rank k is increased, until k reaches the nonnegative rank.

3.2.1 Proposed Algorithm: DC-NMF

The NMF problem shown in (1.1) can be recast as

$$\min_{W\geq 0}\min_{H\geq 0}\|X-WH\|_F^2=\min_{W\geq 0}\left\{\min_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n\in\operatorname{span}_+(W)}\|X-[\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n]\|_F^2\right\}\stackrel{\text{def}}{=}\min_{W\geq 0}e_X(W),$$

where $\operatorname{span}_+(W)\stackrel{\text{def}}{=}\{\sum_{i=1}^{k}h_i\boldsymbol{w}_i|h_i\in\mathbb{R}_+\}$ is the conical hull of $W=[\boldsymbol{w}_1,\ldots,\boldsymbol{w}_k]\in\mathbb{R}_+^{m\times k}$. Accordingly, rank-$k$ NMF can be interpreted as finding a nonnegative basis $\boldsymbol{w}_1,\ldots,\boldsymbol{w}_k$ such that X can be best approximated by vectors in $\operatorname{span}_+(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_k)$. We will use the notation $X\approx\operatorname{span}_+(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_k)$ to denote that X is approximated by the conical hull of nonnegative vectors $\boldsymbol{w}_i$'s.

Unlike for the SVD, one cannot use successive rank-1 deflations to go from rank-2 NMF to rank-k NMF for $k>2$ [34]. For NMF, all vectors in $W\in\mathbb{R}^{m\times k}$ typically change completely when the reduced rank k changes. However, since rank-2 NMF can be used for binary clustering, the columns of X can be divided into two clusters based on H from rank-2 NMF, forming two submatrices X_1 and X_2, as illustrated in Figure 3.2. Assume we have a rank-2 NMF of X as $X\approx\operatorname{span}_+(\boldsymbol{w}_1,\boldsymbol{w}_2)$. Then we view $\boldsymbol{w}_i$ as a representative vector for X_i, i.e., $X_i\approx\operatorname{span}_+(\boldsymbol{w}_i)$, for $i=1,2$. If $\operatorname{rank}_+(X)>2$, then we can obtain a better approximation of X by replacing one of $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ with two basis vectors obtained by applying rank-2 NMF on X_1 or X_2. Our cluster tree traversing rule determines this next submatrix for which we increase the reduced rank for NMF approximation from 1 to 2, so that the overall nonnegative approximation error for X is locally reduced the most. For example, in Figure 3.2, applying rank-2 NMF on X_1 gives us two new submatrices X_{11} and X_{12} where $X_{11}\approx\operatorname{span}_+(\boldsymbol{w}_{11})$ and $X_{12}\approx\operatorname{span}_+(\boldsymbol{w}_{12})$. Then according to

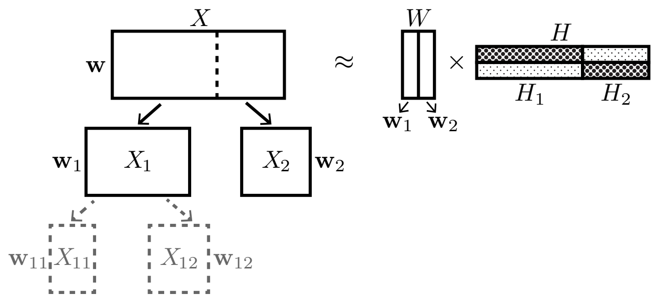


Figure 3.2: Illustration of how DC-NMF use divide-and-conquer to go from rank-2 NMF to higher rank NMF. The dark part in H means relative larger values.

Theorem 3.1, we have $e_{X_1}([\boldsymbol{w}_{11}, \boldsymbol{w}_{12}]) < e_{X_1}(\boldsymbol{w}_1)$. Since the total approximation error for X is controlled by such local errors (see Theorem 3.2), X is better approximated by vectors in $\operatorname{span}_+(\boldsymbol{w}_{11}, \boldsymbol{w}_{12}, \boldsymbol{w}_2)$, which gives us a good rank-3 approximation. We repeat the above divide-and-conquer steps until we reach the desired reduced rank k. The above procedure consists of the following three key components:

S1. We partition a matrix X into two submatrices X_1 and X_2 according to the factor H from the rank-2 NMF of X by the following rule: the j-th column of X belongs to X_1 if $H[1, j] > H[2, j]$; otherwise it belongs to X_2. We then take $\boldsymbol{w}_i$ as a representative vector for X_i, $X_i \approx \operatorname{span}_+(\boldsymbol{w}_i)$, $i = 1, 2$. We denote this procedure as

$$(X_1, X_2, \boldsymbol{w}_1, \boldsymbol{w}_2) = \text{SPLIT}(X).$$

S2. Suppose matrix $X \approx \operatorname{span}_+(\boldsymbol{w})$ in step **S1**. We measure the effect of the reduced approximation error from the increase of the reduced rank from 1 to 2 for X by the score

$$e_X(\boldsymbol{w}) - (e_{X_1}(\boldsymbol{w}_1) + e_{X_2}(\boldsymbol{w}_2)) = \min_{\boldsymbol{h}} \|X - \boldsymbol{w}\boldsymbol{h}^\intercal\|_F^2 - \sum_{i=1}^{2} \min_{\boldsymbol{h}_i} \|X_i - \boldsymbol{w}_i\boldsymbol{h}_i^\intercal\|_F^2.$$

We denote this procedure as

$$\mathit{score} = \text{COMPUTE\_SCORE}(X, X_1, X_2, \boldsymbol{w}, \boldsymbol{w}_1, \boldsymbol{w}_2).$$

Note that the solutions $\boldsymbol{h}$ and $\boldsymbol{h}_i$ will be automatically nonnegative since X, X_i, $\boldsymbol{w}$, $\boldsymbol{w}_i$ are all nonnegative.

S3. We recursively apply Step **S1** $(k-1)$ times, dividing columns of X into k clusters and obtaining one representative vector for each cluster, resulting in k vectors in total, which are the column vectors of the desired $W \in \mathbb{R}_+^{m\times k}$. Each time we apply **S1**, we choose to further split the submatrix that will result in the largest approximation error decrease measured by the local score defined in **S2**. This step is described in detail in Algorithm 3. In Algorithm 3, each X_i is represented by $\boldsymbol{w}_i$ and we use $W = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k]$ to represent X.

Theorem 3.2. *Suppose $X \in \mathbb{R}_+^{m\times n}$, and the columns of X are partitioned into $[X_1, \ldots, X_k]$ with $2 \le k \le n$ and $X_i \in \mathbb{R}_+^{m\times n_i}$. For any $W = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k] \in \mathbb{R}_+^{m\times k}$, we have $e_X(W) \le \sum_{i=1}^k e_{X_i}(\boldsymbol{w}_i)$, i.e.*

$$\min_{H\in\mathbb{R}_+^{k\times n}} \|X - WH\|_F^2 \le \sum_{i=1}^{k} \min_{\boldsymbol{h}_i\in\mathbb{R}_+^{n_i\times 1}} \|X_i - \boldsymbol{w}_i\boldsymbol{h}_i^\intercal\|_F^2.$$

Proof. Let $\widehat{\boldsymbol{h}}_i = \arg\min_{\boldsymbol{h}} \|X_i - \boldsymbol{w}_i\boldsymbol{h}^\intercal\|_F^2$ and $\widehat{H} = [\widehat{H}_1, \ldots, \widehat{H}_k]$, where $\widehat{H}_i \in \mathbb{R}_+^{k\times n_i}$ has $\widehat{\boldsymbol{h}}_i^\intercal$ as its ith row and zeros in all other entries. Then, $\min\limits_{H} \|X - WH\|_F^2 \le \|X - W\widehat{H}\|_F^2 = \sum\limits_{i=1}^{k} \|X_i - W\widehat{H}_i\|_F^2 = \sum\limits_{i=1}^{k} \|X_i - \boldsymbol{w}_i\widehat{\boldsymbol{h}}_i^\intercal\|_F^2 = \sum\limits_{i=1}^{k} \min\limits_{\boldsymbol{h}_i} \|X_i - \boldsymbol{w}_i\boldsymbol{h}_i^\intercal\|_F^2.$ □

The matrix X (after a proper permutation of columns), the partition $X_1, \ldots, X_k$ and the representative vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ from Algorithm 3 satisfy the conditions in Theorem 3.2. This means that if we use the W collected from Algorithm 3 and obtain H from one step of NLS $\min_{H\ge 0} \|WH - X\|_F$ to get W and H as the NMF solution, the approximation error $\|X - WH\|_F^2$ will be bounded by $\sum_{i=1}^k e_{X_i}(\boldsymbol{w}_i)$, which is what we minimize in each step of Algorithm 3. We can also perform several more NLS iterations for NMF to further reduce the approximation error using W from Algorithm 3 as the initial guess for W. The stopping criteria for flat NMF can be used here, for example the one used in [31] that checks whether

Algorithm 3 Algorithm to generate basis vectors $W = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k] \in \mathbb{R}_+^{m \times k}$ for an input matrix $X \in \mathbb{R}_+^{m \times n}$ and reduced dimension $k > 2$ based on Rank-2 NMF and tree-traversing rule.

Input: $X \in \mathbb{R}_+^{m \times n}$, $k > 2$
Output: $W = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k] \in \mathbb{R}_+^{m \times k}$

$X_1 \leftarrow X$, $\mathrm{score}(X_1) \leftarrow \infty$
$(X_{11}, X_{12}, \boldsymbol{w}_{11}, \boldsymbol{w}_{12}) \leftarrow \textsc{split}(X_1)$
for $l = 2 : k$ **do**
  $j \leftarrow \underset{1 \le i < l}{\arg\max}\ \mathrm{score}(X_i)$
  $X_j \leftarrow X_{j1}$, $\boldsymbol{w}_j \leftarrow \boldsymbol{w}_{j1}$, $X_l \leftarrow X_{j2}$, $\boldsymbol{w}_l \leftarrow \boldsymbol{w}_{j2}$
  if $l < k$ **then**
    $(X_{j1}, X_{j2}, \boldsymbol{w}_{j1}, \boldsymbol{w}_{j2}) \leftarrow \textsc{split}(X_j)$
    $\mathrm{score}(X_j) \leftarrow \textsc{compute\_score}(X_j, X_{j1}, X_{j2}, \boldsymbol{w}_j, \boldsymbol{w}_{j1}, \boldsymbol{w}_{j2})$
    $(X_{l1}, X_{l2}, \boldsymbol{w}_{l1}, \boldsymbol{w}_{l2}) \leftarrow \textsc{split}(X_l)$
    $\mathrm{score}(X_l) \leftarrow \textsc{compute\_score}(X_l, X_{l1}, X_{l2}, \boldsymbol{w}_l, \boldsymbol{w}_{l1}, \boldsymbol{w}_{l2})$
  end if
end for

the solution is a stationary point. In practice, we have found that there is usually a significant drop of approximation error after one full alternating iteration of computing H and then updating W, and subsequent iterations did not significantly reduce the approximation error. Therefore, in our proposed DC-NMF, we perform one iteration to compute H and update W starting with W given by Algorithm 3, in order to obtain a good solution while maintaining the speed advantage. The approximation error $\|X - WH\|_F^2$ can be computed by the formula $\|X - WH\|_F^2 = \|X\|_F^2 - 2 \cdot \mathrm{trace}(HX^\intercal W) + \mathrm{trace}(W^\intercal WHH^\intercal)$ to avoid directly computing $X - WH$, which is computationally expensive and can destroy the sparse structure of X.

3.2.2 Other Possibilities for DC-NMF

The priority scores for DC-NMF proposed in [10, 12] need to pre-split a cluster (of columns) in order to compute a priority score. We can also define heuristic scores that do not need a pre-split. For example, supposing $\tilde{X}$ is a submatrix corresponding to a cluster and $\tilde{\boldsymbol{w}}$ is its representative vector, we can define a heuristic score as $\min_{\tilde{\boldsymbol{h}}} \|\tilde{X} - \tilde{\boldsymbol{w}}\tilde{\boldsymbol{h}}^\intercal\|_F$ to check how well $\tilde{X}$ is represented as rank-1 matrix $\tilde{\boldsymbol{w}}\tilde{\boldsymbol{h}}^{\intercal}$ i.e., how coherent its columns are, and split (i.e. approximate by rank 2) the worst represented cluster.

To describe these priority scores in a unified way, we use the notations as shown in Figure 3.2, with tilde added to each symbol, such that $\tilde{X} \approx \mathrm{span}_{+}(\tilde{\boldsymbol{w}})$, $\tilde{X}_i \approx \mathrm{span}_{+}(\tilde{\boldsymbol{w}}_{\boldsymbol{i}})$ ($i = 1, 2$), where $\tilde{X}$ is a submatrix of the original data matrix X, consisting of a cluster of columns of X. After one step of rank-2 NMF, the matrix $\tilde{X}$ is divided into two submatrices $\tilde{X}_1$ and $\tilde{X}_2$. For methods that do not need pre-split, we can directly compute priority score s_1, s_2 for $\tilde{X}_1$ and $\tilde{X}_2$, using $\tilde{\boldsymbol{w}}_1$ and $\tilde{\boldsymbol{w}}_2$, respectively. However, for methods that need pre-split, we can only compute the priority score s for $\tilde{X}$ with the same information. We summarize some of the priority scores in Table 3.2, where $\tilde{\boldsymbol{h}}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ are column vectors of proper size.

In our experiments, we found that Score 0 and Score 1 often obtain significantly lower approximation errors than the other scores. However, Score 1 requires significantly longer computation time than Score 0 since Score 1 also computes a rank-1 SVD. Our tests show that when we start two DC-NMF computations with Score 0 and Score 1 at the same time, by the time DC-NMF with Score 1 completes computation of W from Algorithm 3, DC-NMF with Score 0 completes computation of an initial W and runs several alternating NLS iterations for NMF, obtaining better solutions than DC-NMF with Score 1. Therefore, we recommend Score 0 in practice.

3.3 Experiments

In this section, we show experimental results for DC-NMF and compare it with state-of-the-art algorithms for NMF, clustering, and topic modeling. First, we focus on the role of DC-NMF as a generic algorithm for computing NMF and evaluate its runtime versus approximation error. Then, we apply DC-NMF to small- to medium-scale data sets with ground-truth to evaluate its effectiveness for clustering before moving to much larger data sets for the benchmarking of computational efficiency. Our experiments were run on a server

Table 3.2: Various priority scores for choosing a cluster to split.

Name	Formula	Need Pre-split	Note
Score 0	$s = \min_{\tilde{\boldsymbol{h}}} \|\tilde{X} - \tilde{\boldsymbol{w}}\tilde{\boldsymbol{h}}^\intercal\|_F^2 - \sum_{i=1}^{2} \min_{\tilde{\boldsymbol{h}}_i} \|\tilde{X}_i - \tilde{\boldsymbol{w}}_i\tilde{\boldsymbol{h}}_i^\intercal\|_F^2$	Y	The score proposed in this chapter
Score 1	$s = \min_{\boldsymbol{u},\boldsymbol{v}} \|\tilde{X} - \boldsymbol{u}\boldsymbol{v}^\intercal\|_F^2 - \sum_{i=1}^{2} \min_{\boldsymbol{u}_i,\boldsymbol{v}_i} \|\tilde{X}_i - \boldsymbol{u}_i\boldsymbol{v}_i^\intercal\|_F^2$	Y	The score used for hierarchical clustering in [12]
Score 2	$s = \mathrm{mNDCG}(\tilde{\boldsymbol{w}}_1) \times \mathrm{mNDCG}(\tilde{\boldsymbol{w}}_2)$	Y	The score used for hierarchical topic modeling in [10]
Score 3	$s_i = \min_{\tilde{\boldsymbol{h}}} \|\tilde{X}_i - \tilde{\boldsymbol{w}}_i\tilde{\boldsymbol{h}}^\intercal\|_F^2$	N	Measures how well $\tilde{X}_i$ is represented by $\tilde{\boldsymbol{w}}_i$.
Score 4	$s_i = \min_{\boldsymbol{u},\boldsymbol{v}} \|\tilde{X}_i - \boldsymbol{u}\boldsymbol{v}^\intercal\|_F^2$	N	Measures how close $\tilde{X}_i$ is to a rank-1 matrix.
Score 5	$s_i = \|\tilde{X}_i - \tilde{W}\tilde{H}_i\|_F^2$	N	Measures how well $\tilde{X}_i$ is represented by $\tilde{W}$.

with two Intel E5-2620 processors, each having six cores, and 377 GB memory.

Before proceeding to the experimental results, we first describe the data sets and experimental settings in detail.

3.3.1 Data Sets

Six text data sets were used in our experiments: 1. **Reuters-21578**[1] contains news articles from the Reuters newswire in 1987. We discarded documents with multiple class labels, and then selected the 20 largest classes. 2. **20 Newsgroups**[2] contains articles from Usenet newsgroups and have a defined hierarchy of 3 levels. Usenet users post messages and reply to posts under various discussion boards, often including a personalized signature at the end of their messages. Unlike the widely-used indexing of this data set[2], we observed that many articles had duplicate paragraphs due to cross-referencing. We discarded cited paragraphs and signatures, which increased the difficulty of clustering. 3. **Cora** [42] is a collection of research papers in computer science, from which we extracted the title, abstract, and reference-contexts. Although this data set comes with a predefined topic hierarchy of

[1] `http://www.daviddlewis.com/resources/testcollections/reuters21578/` (retrieved in June 2014)

[2] `http://qwone.com/~jason/20Newsgroups/` (retrieved in June 2014)

Table 3.3: Data sets used in our experiments. Numbers in parentheses are the numbers of clusters/topics we requested for unlabeled data sets.

Data sets	Has label	Has hierarchy	# terms	# docs	# nodes at each level
Reuters-21578	Y	N	12,411	7,984	20
20 Newsgroups	Y	Y	36,568	18,221	6/18/20
Cora	Y	N	154,134	29,169	70
NIPS	Y	N	17,981	447	13
RCV1	N	-	149,113	764,751	(60)
Wiki-4.5M	N	-	2,361,566	4,126,013	(80)

3 levels, we observed that some topics, such as "AI – NLP" and "IR – Extraction", were closely related but resided in different subtrees. Thus, we ignored the hierarchy and obtained 70 ground-truth classes as a flat partitioning. 4. **NIPS** is a collection of NIPS conference papers. We chose 447 papers from the 2001–2003 period [43], which were associated with labels indicating the technical area (algorithms, learning theory, vision science, etc). 5. **RCV1** [44] is a much larger collection of news articles from Reuters, containing about 800,000 articles from the time period of 1996–1997. We used the entire collection as an unlabeled data set. 6. **Wikipedia**[3] is an online, user-contributed encyclopedia and provides periodic dumps of the entire website. We processed the dump of all the English Wikipedia articles from March 2014, and used the resulting 4.5 million documents as an unlabeled data set **Wiki-4.5M**, ignoring user-defined categories.

We summarize these data sets in Table 3.3. The first four medium-scale data sets have ground-truth labels for the evaluation of cluster quality, while the remaining two large scale data sets are treated as unlabeled. All the labeled data sets except 20 Newsgroups have very unbalanced sizes of ground-truth classes. We constructed the normalized-cut weighted version of term-document matrices as in [1].

[3] `https://dumps.Wikimedia.org/enWiki/`

3.3.2 Implementation

We implemented DC-NMF both in Matlab and in an open-source C++ software library called `SmallK`[4] [5]. The existing methods we compared DC-NMF with are grouped into three categories: NMF algorithms, clustering methods and topic modeling methods. Though clustering and topic modeling can be unified in the framework of matrix factorization, as explained in Section 1.2.2, we label a method as belonging to one of the two categories according to the task for which it was originally targeted.

NMF Algorithms. We compared the following algorithms for computing rank-k NMF:[5]

- `MU`: The multiplicative update algorithm for Frobenius-norm based NMF [7]. `MU` is not guaranteed to converge to a stationary point solution although it reduces the objective function after each iteration.

- `ANLS/BPP`: The block principal pivoting algorithm that follows the two-block coordinate descent framework [46, 31]. We will often refer to this method as simply BPP.

- `HALS/RRI`: The hierarchical alternating least squares algorithm [47, 48], which is a $2k$-block coordinate descent method. We will simply refer to this as `HALS`.

Many schemes that can be used to accelerate the above algorithms have been proposed in the literature (e.g. [49, 50]) but our comparisons will be on the above baseline algorithms.

Clustering Methods. The clustering methods we compared include:

- `nmf-hier`: Hierarchical clustering based on standard NMF with ANLS and an active-set method for NLS [51]. The active-set method searches through the space of active-set/passive-set partitionings for the optimal active set, with a strategy that reduces the objective function at each search step.

---

[4] `https://smallk.github.io/`

[5] Besides the listed algorithms, we also experimented with a recent algorithm based on coordinate descent with a greedy rule to select the variable to improve at each step [45]. However, this algorithm became increasingly slow when we increased k and kept the size of A the same. Therefore, we did not include it in our final comparison.

- nmf-flat: Flat clustering based on standard NMF with ANLS. The block principal pivoting (BPP) method [46, 31] is used as an exemplar algorithm to solve the NLS subproblems. In our experiments, multiplicative update rule algorithms [2] were always slower and gave similar quality compared to active-set-type algorithms, thus were not included in our results.

- kmeans-hier: Hierarchical clustering based on standard K-means. We used the hierarchical clustering workflow described in [10].

- kmeans-flat: Flat clustering based on standard K-means.

- CLUTO: A clustering toolkit[6] written in C++. We used the default method in its vcluster program, namely a repeated bisection algorithm.

Topic Modeling Methods. The topic modeling methods we compared include:

- Mallet-LDA: The software MALLET[7] written in Java for flat topic modeling, which uses the Gibbs sampling algorithm for LDA. 1000 iterations were used by default.

- AnchorRecovery: A recent fast algorithm to solve NMF with separability constraints [52]. It selects an "anchor" word for each topic, for example, "Los Angeles Clippers" rather than "basketball", which could carry a narrow meaning and not semantically represent the topic [52]. The software is written in Java[8]. We used the default parameters.

- XRAY: Another recent algorithm to solve NMF with separability constraints [53]. It incrementally selects "extreme rays" to find a cone that contains all the data points. We used the *greedy* option as the selection criteria in this algorithm.

- Hottopixx: A recent method that formulates Separable NMF as a linear program and solves it using incremental gradient descent [54]. We used the default parameters.

[6] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview
[7] http://mallet.cs.umass.edu/
[8] https://github.com/mimno/anchor

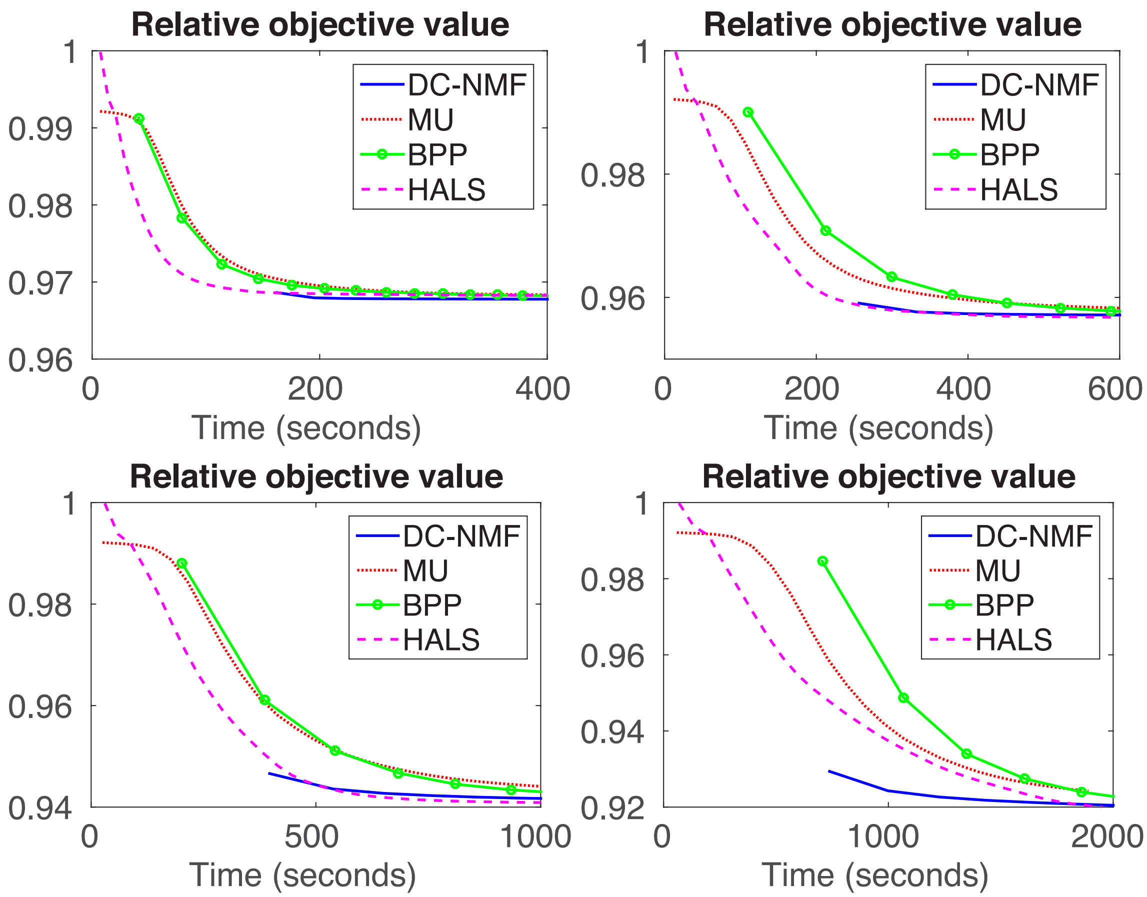


Figure 3.3: Comparison of approximation error between DC-NMF versus other algorithms for computing NMF. Results are shown for $k = 20, 40, 80, 160$.

3.3.3 Experimental Settings

To evaluate the cluster and topic quality, we use the *normalized mutual information* (NMI). For data sets with defined hierarchy, we compute NMI between a generated partitioning and the ground-truth classes at each level of the ground-truth tree. In other words, if the ground-truth tree has depth L, we compute L NMI measures, one for each level. When evaluating the results given by DC-NMF (Algorithm 3), we treat all the outliers as one separate cluster for fair evaluation.

Hierarchical clusters and flat clusters cannot be compared against each other directly. When evaluating the hierarchical clusters, we take snapshots of the tree as leaf nodes are generated, and treat all the leaf nodes in each snapshot as a flat partitioning which is to

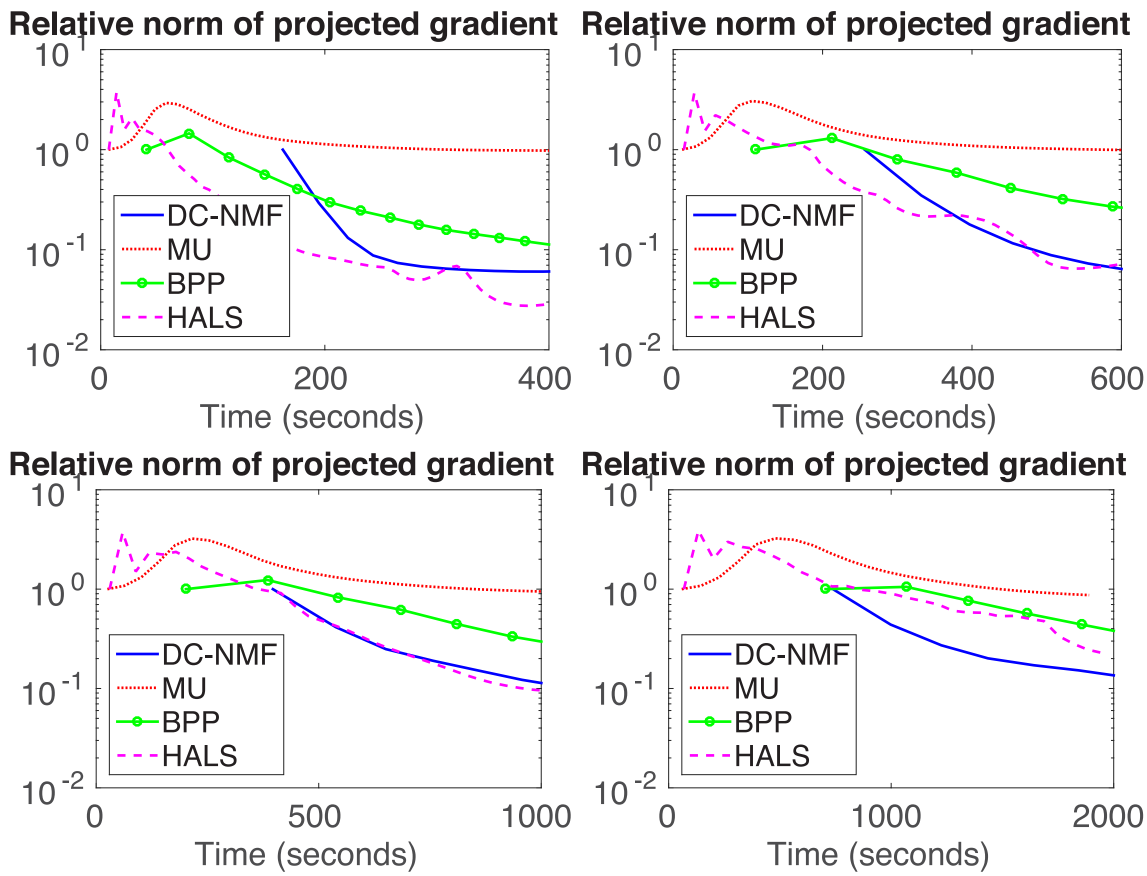


Figure 3.4: Comparison of projected gradient norm between DC-NMF versus other algorithms for computing NMF. Results are shown for $k = 20, 40, 80, 160$.

be compared against the ground-truth classes. This is possible since the leaf nodes are non-overlapping. Thus, if the maximum number of leaf nodes is set to c, we produce $c - 1$ flat partitionings forming a hierarchy. For each method, we perform 20 runs with random initializations. Average measurements are reported. Note that for flat clustering methods, each run consists of $c - 1$ separate executions with the number of clusters set to $2, 3, \cdots, c$.

The maximum number of leaf nodes c is set to be the number of ground-truth labels at the deepest level for labeled data sets (see Table 3.3); and we set $c = 60$ for **RCV1** and $c = 80$ for **Wiki-4.5M**. The Matlab `kmeans` function has a batch update phase and a more time consuming online update phase. We rewrote this function using BLAS-3 operations and boosted its efficiency substantially[9]. We use both phases for data sets with fewer than

[9] `http://math.ucla.edu/~dakuang/software/kmeans3.html`

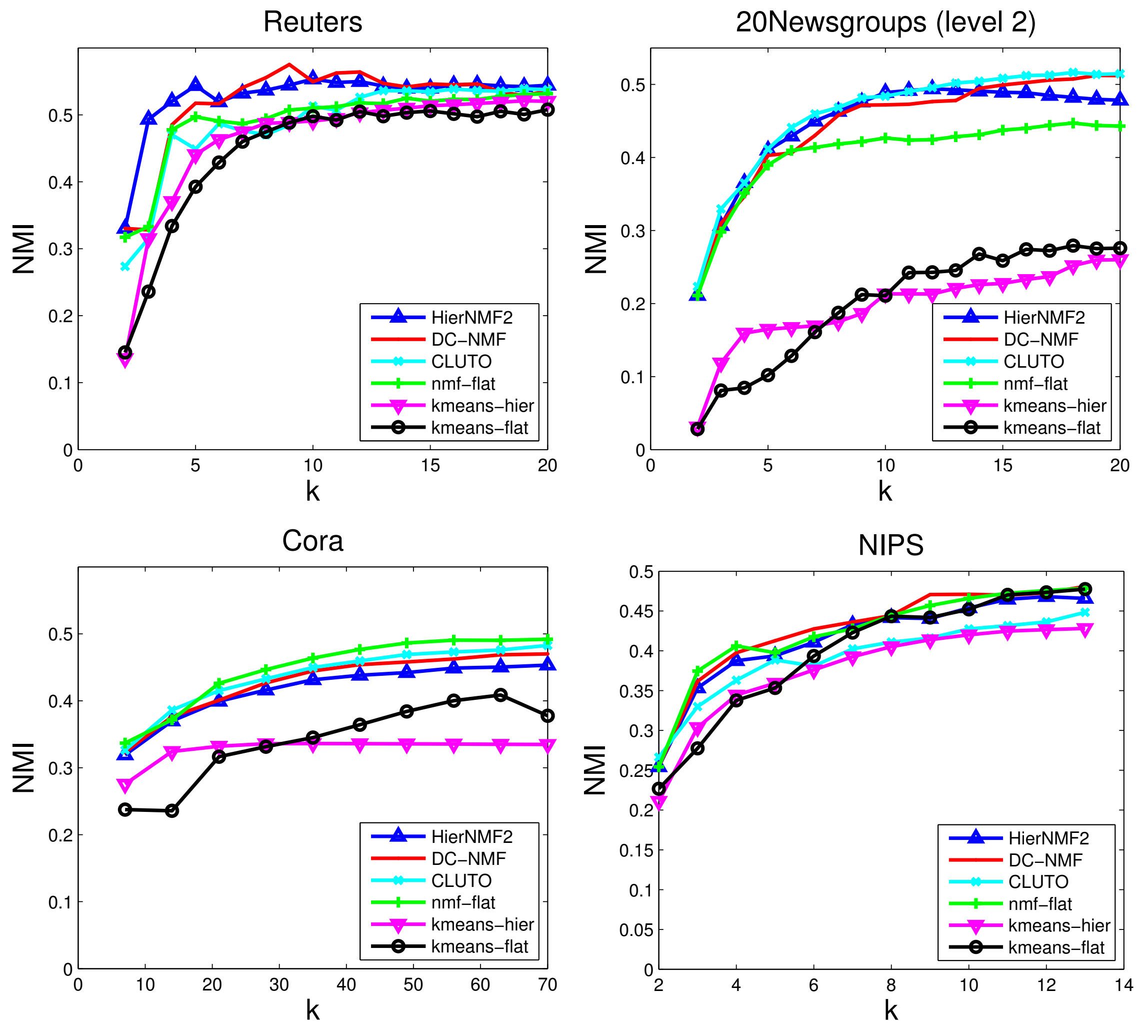


Figure 3.5: DC-NMF versus other *clustering methods* in cluster quality evaluated by normalized mutual information (NMI).

20,000 documents, and only the batch-update phase for data sets with more than 20,000 documents. For NMF, we use the projected gradient norm as the stopping criterion [55] with a tolerance parameter $\epsilon = 10^{-4}$. The projected gradient norm is sensitive to the scaling of the W and H factors: WD and $D^{-1}H$ yield the same approximation error but different values of projected gradient norm, where D is a diagonal matrix with positive entries on the diagonal (see details in [31, 56]). To ensure a fair comparison between different methods, before computing a projected gradient norm, we make the columns of W have unit 2-norm and scale H accordingly. All the methods are implemented with multi-threading.

3.3.4 DC-NMF for Computing Rank-k NMF

Since our focus in this chapter is on large-scale NMF, we compare the algorithms for computing rank-k NMF on a large-scale text data set, namely RCV1. We report the relative approximation error (i.e. $\|X - WH\|_F / \|X\|_F$) and relative projected gradient norm (i.e. projected gradient norms divided by the initial projected gradient norm) achieved by each algorithm in Figure 3.3 and Figure 3.4. While the approximation error measures the effectiveness of an NMF algorithm, the projected gradient norm determines when to stop the algorithm, and is thus important for the run-time in actual use of these algorithms. The results show that HALS and DC-NMF produce the smallest approximation error and projected gradient norm, and DC-NMF has more advantages when k increases. Note that HALS may run into problems of divide-by-zero and we also found that the results were very sensitive to the way zeros were treated numerically. DC-NMF algorithm does not have such a problem nor require any parameters.

3.3.5 DC-NMF for Clustering and Topic Modeling

Cluster Quality

Figs. 3.5 and 3.6 show the cluster quality on four labeled data sets, comparing DC-NMF with the state-of-the-art *clustering methods* and *topic modeling methods*, respectively. `nmf-hier` generates the identical results with DC-NMF (but the former is less efficient) and is not shown in Figure 3.5.

We can see that DC-NMF gives better cluster and topic quality in many cases, and improves the performance of HierNMF2 in every case. One possible reason for the better performance of DC-NMF is that documents that appear to be outliers are removed when building the hierarchy in HierNMF2, and thus the topics at the leaf nodes are more meaningful and represent more salient topics than those generated by a flat topic modeling method that takes every document into account. The algorithms solving NMF with separability

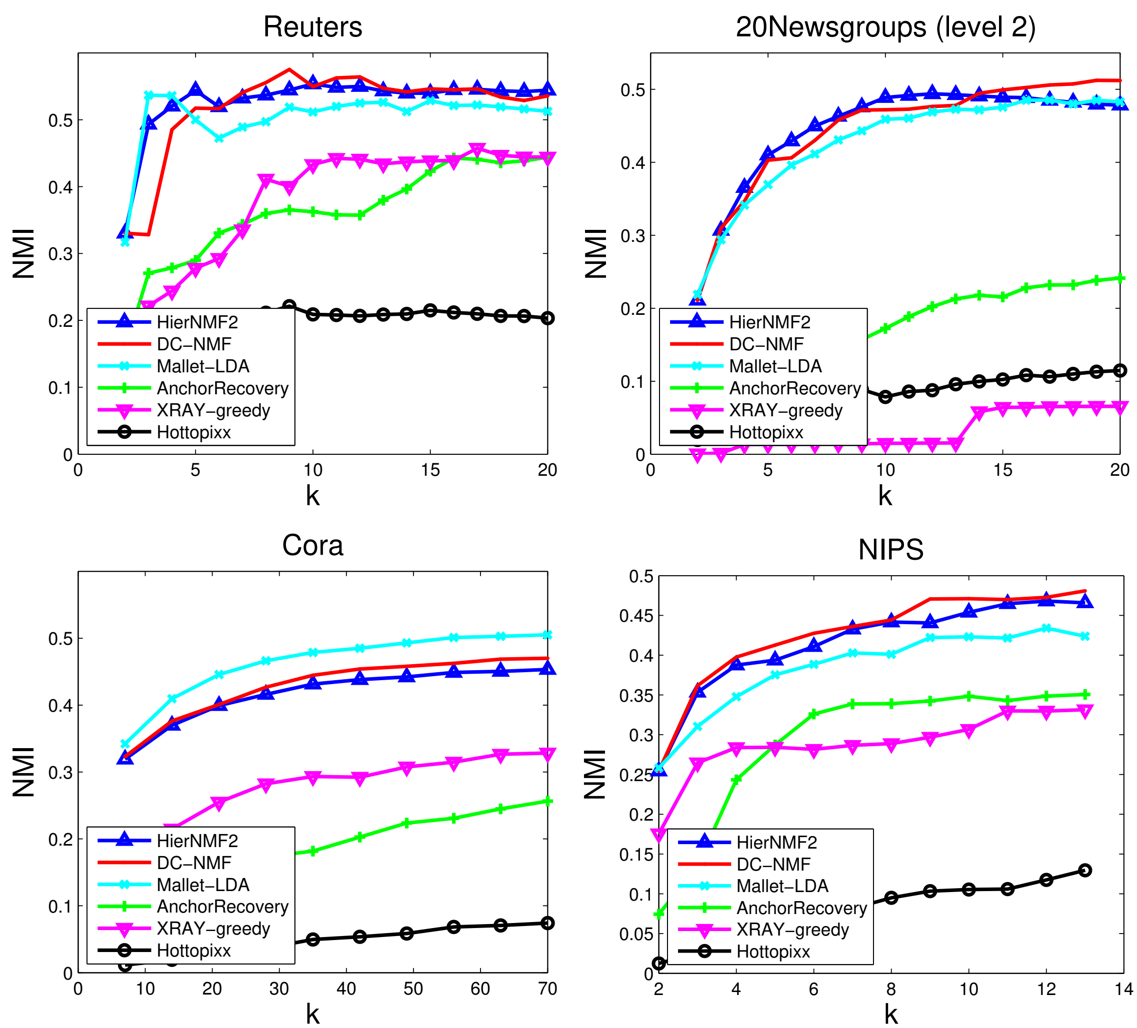


Figure 3.6: DC-NMF versus other *topic modeling methods* in cluster quality evaluated by normalized mutual information (NMI).

constraints yielded the lowest clustering quality. Among them, `AnchorRecovery` and `Hottopixx` both require several parameters provided by the user, which could be time-consuming to tune and have a large impact on the performance of their algorithms. We used the default parameters for both of these methods, which may have negatively affected their NMIs.

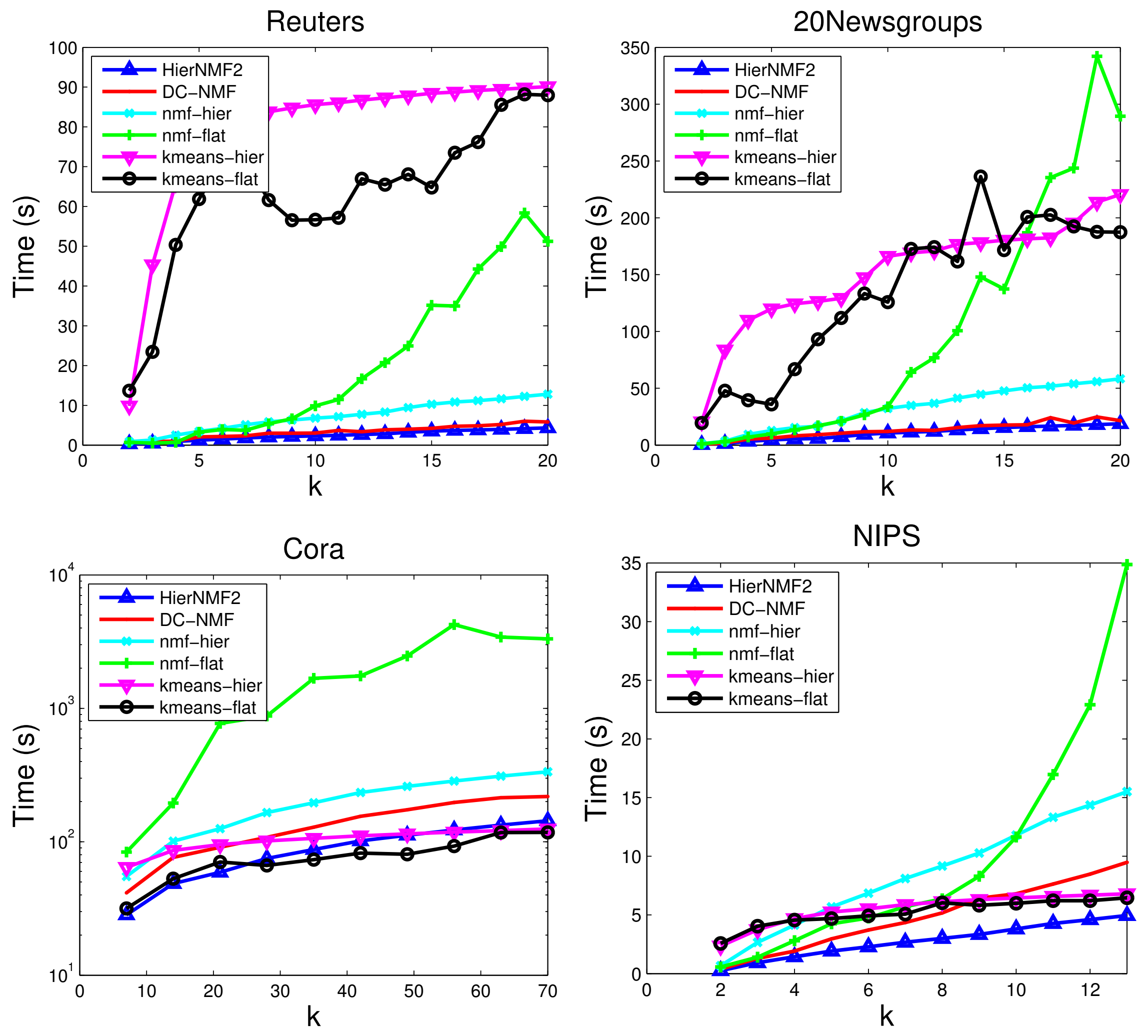


Figure 3.7: Timing results for the Matlab implementation of HierNMF2, DC-NMF, NMF, and K-means on the smaller data sets.

Timing Results

Figure 3.7 shows the run-time of the proposed methods versus NMF and K-means, all implemented in Matlab. DC-NMF required substantially less run-time compared to the standard flat NMF. These results show that flat clustering based on standard NMF exhibits a superlinear trend while hierarchical clustering based on Rank-2 NMF exhibits a linear trend of runtime as k increases. For example, to generate 70 clusters on the **Cora** data set, HierNMF2, DC-NMF, `nmf-hier`, and `nmf-flat` took about 2.4, 2.6, 5.6, and 55.3 minutes, respectively. We note that K-means with only the batch-update phase has similar

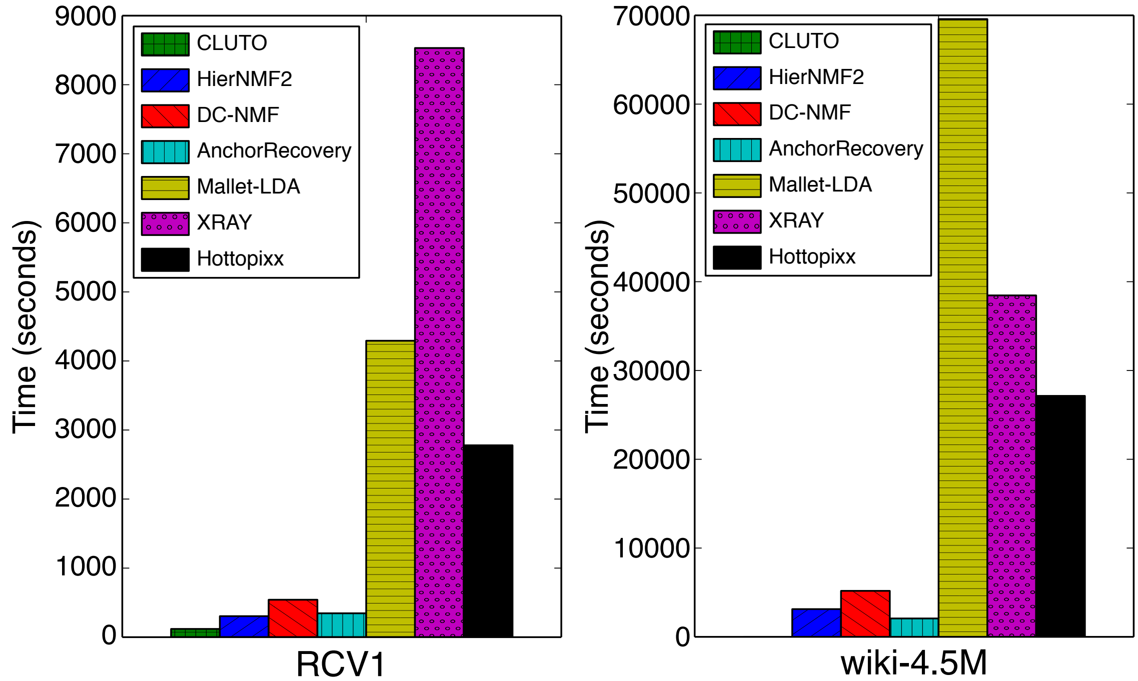


Figure 3.8: Timing results for the C++ implementation of HierNMF2 and DC-NMF available in our open-source software `smallk` and other state-of-the-art clustering and topic modeling methods on large, unlabeled text data sets.

runtime to DC-NMF; however, the cluster quality is not as good, which was shown earlier in Figure 3.5.

Figure 3.8 compares the run-time of our C++ implementation of DC-NMF available in the software `smallk` [5] versus off-the-shelf toolkits (`CLUTO`, `Mallet-LDA`) and recent methods proposed for large-scale topic modeling, namely `AnchorRecovery`, `XRAY`, and `Hottopixx`. We used 8 threads when possible to set the number of threads manually (in the cases of `smallk`, `CLUTO`, `Mallet-LDA`, and `Hottopixx`).

On the **RCV1** and **Wiki-4.5M** data sets, DC-NMF is about 20 times faster than `Mallet-LDA`; particularly on the largest **Wiki-4.5M** data set in our experiments, DC-NMF found 80 topics in about 50 minutes, greatly enhancing the practicality of topic modeling algorithms when compared to the other software packages in our experiments.

The three algorithms `AnchorRecovery`, `XRAY`, and `Hottopixx` that solve NMF with separability constraints require a large $m \times m$ matrix, i.e. word-word similarities. We reduced the vocabulary of **Wiki-4.5M** to about 100,000 unique terms in order to accommodate the $m \times m$ matrix in main memory for these algorithms. Among them, `XRAY` and `Hottopixx` build a dense word-word similarity matrix and thus have a large memory footprint [53, 54]. `AnchorRecovery`, on the other hand, computes a random projection of the word-word similarity matrix, greatly reducing the time and space complexity [52]; however, as we have seen in Figure 3.6, its cluster quality is not as good as that of DC-NMF.

Overall, DC-NMF is the best-performing method in our experiments, considering both cluster quality and efficiency. The relatively recent software package `CLUTO` is also competitive.[10]

3.3.6 Illustration

To visualize the cluster/topic tree generated by HierNMF2, we show an illustration of the topic structure for a news article data set containing 100,361 articles in Figure 3.9. First, we notice that the tree was not restrained to have a balanced structure, and HierNMF2 was able to determine the semantic organization on-the-fly. We can see that the articles were first divided into two big categories—politics/economy and art/entertainment/life. In the next few hierarchical levels, those topics (politics, economy, art, etc.) were further refined and emerged as more coherent sub-topics. Finally, at the leaf level, HierNMF2 produced fine-grained topics such as Iraq war, law and justice, stock market, movies, musics, health, houses and hotels.

[10]The run-time for `CLUTO` on **Wiki-4.5M** is absent: on our smaller system with 24 GB memory, it ran out of memory; and on our larger server with sufficient memory, the binary could not open a large data file (> 6 GB). The `CLUTO` software is not open-source and thus we only have access to the binary and are not able to build the program on our server.

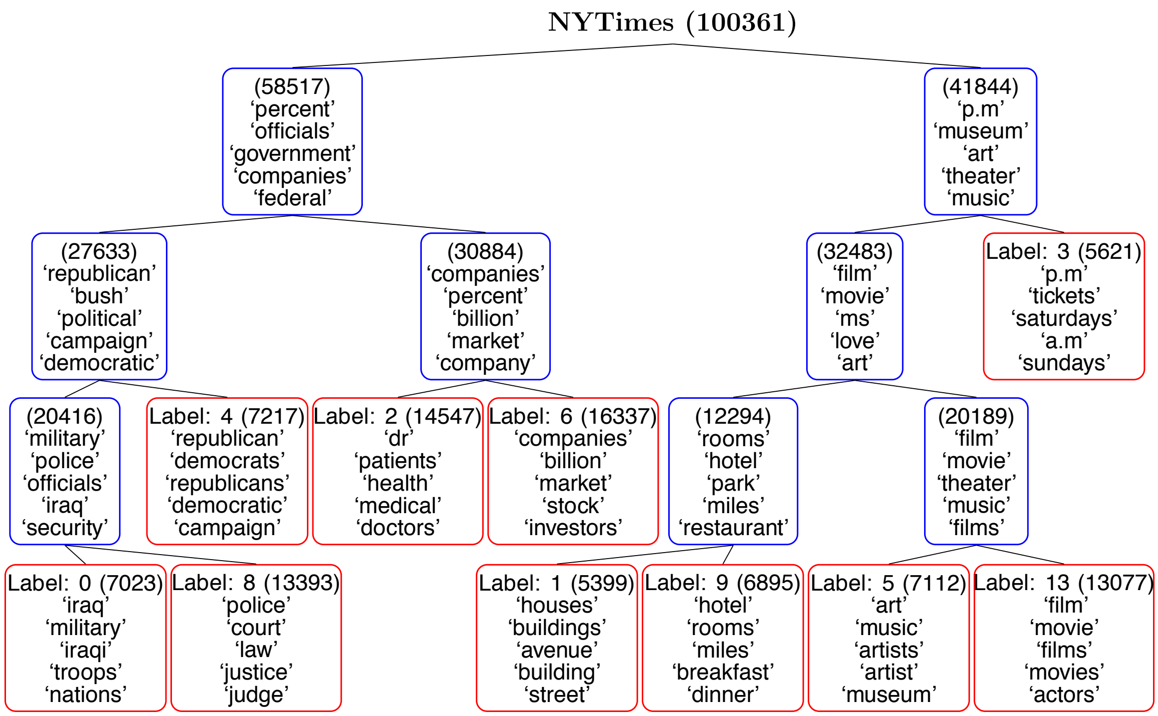


Figure 3.9: Hierarchical clustering result generated on a data set consisting of 100,361 New York Times articles for illustration. The hierarchy is automatically detected and not necessarily a balanced tree. Each tree node $\mathcal{N}$ is associated with a column of W, denoted as $\mathbf{w}_{\mathcal{N}}$, generated by Rank-2 NMF applied on its parent node. We display the five terms with highest importance values in $\mathbf{w}_{\mathcal{N}}$. Red boxes indicate leaf nodes while blue boxes indicate non-leaf nodes. The number in the parentheses at each node indicates the number of documents associated with that node.

CHAPTER 4

HIERSYMNMF2 FOR LARGE SCALE SIMILARITY/CONNECTION-BASED CLUSTERING

4.1 SymNMF for Similarity/Connection-Based Clustering

As discussed in Section 1.2.1, similarity/connection information can usually be encoded in a nonnegative symmetric matrix. To utilize the symmetric structure, we need a variant of NMF called Symmetric NMF (SymNMF) [11], the formulation of which is

$$\min_{H \geq 0} \|S - H^{\intercal} H\|_F, \tag{4.1}$$

where $S \in \mathbb{R}_+^{n \times n}$ is a nonnegative symmetric matrix, $H \in \mathbb{R}_+^{k \times n}$ and $k \ll n$. Here we assume S is the matrix representation of a graph since most similarity/connection relation can be modeled by a graph, as discussed in Section 1.2.1, and the rest of the chapter will use the language of graph clustering or community detection. We assume the graph under study is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. The matrix $S = (w_{ij}) \in \mathbb{R}_+^{n \times n}$ is the adjacency matrix of graph $\mathcal{G}$, where $w_{ij} = w(v_i, v_j)$ is the weight of edge (v_i, v_j). Some choices of the input matrix S for SymNMF are the adjacency matrix $S^{\mathcal{G}}$ and the normalized adjacency matrix $D^{-1/2} S^{\mathcal{G}} D^{-1/2}$, where $D = \operatorname{diag}(d_1, \ldots, d_n)$ and $d_i = \sum_{j=1}^{n} S_{ij}^{\mathcal{G}}$ is the degree of node i. When S is the adjacency matrix, (4.1) is a relaxation of maximizing the ratio association; when S is the normalized adjacency matrix, (4.1) is a relaxation of minimizing the normalized cut [57]. We prove these two relations in the following two theorems, where we will use the convenient Iverson bracket [58]:

$$[\mathcal{P}] = \begin{cases} 1 & \text{if } \mathcal{P} \text{ is true;} \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

where $\mathcal{P}$ is any statement.

Theorem 4.1. *When $S = S^{\mathcal{G}}$, (4.1) is a relaxation of maximizing the ratio association, defined as*

$$\operatorname{rassoc}(P_1, \ldots, P_k) = \sum_{i=1}^{k} \frac{\operatorname{within}(P_i)}{|P_i|}, \tag{4.3}$$

where $(P_1, \ldots, P_k)$ is a partition of $\mathcal{V}$, $|P_i|$ is the number of nodes in P_i, and $\operatorname{within}(P_i)$ is as defined in Section 1.2.3.

Proof. We rewrite Equation (4.3) as

$$\operatorname{rassoc}(P_1, \ldots, P_k) = \sum_{i=1}^{k} \frac{1}{|P_i|} \sum_{u,v \in P_i} w(u, v). \tag{4.4}$$

Let $H = (h_{ij})_{k \times n}$ where n is the number of nodes in the graph, k is the number of partitions and

$$h_{ij} = \frac{1}{\sqrt{|P_i|}} [v_j \in P_i]. \tag{4.5}$$

Then we have

$$\begin{aligned}
\operatorname{tr}(HSH^{\intercal}) &= \sum_{\substack{1 \le i \le k \\ 1 \le j, l \le n}} h_{ij} w_{jl} h_{il} \\
&= \sum_{\substack{1 \le i \le k \\ 1 \le j, l \le n}} \frac{w_{jl}}{|P_i|} [v_j \in P_i][v_l \in P_i] \\
&= \sum_{i=1}^{k} \frac{1}{|P_i|} \sum_{u,v \in P_i} w(u, v) \\
&= \operatorname{rassoc}(P_1, \ldots, P_k),
\end{aligned}$$

and

$$\begin{aligned}(HH^\intercal)_{ij} &= \sum_k h_{ik}h_{jk} \\ &= \sum_k \frac{1}{\sqrt{|P_i||P_j|}}[v_k \in P_i][v_k \in P_j] \\ &= \frac{[i=j]}{|P_i|}\sum_k [v_k \in P_i] \\ &= [i=j],\end{aligned}$$

which means $HH^\intercal = I$. Therefore [59],

$$\begin{aligned}\max \operatorname{rassoc}(P_1, \ldots, P_k) &\Leftrightarrow \max \operatorname{tr}(HSH^\intercal) \\ &\Leftrightarrow \min\{\operatorname{tr}(S^\intercal S) - 2\operatorname{tr}(HSH^\intercal) + \operatorname{tr}(I)\} \\ &\Leftrightarrow \min \operatorname{tr}\left((S - H^\intercal H)^\intercal (S - H^\intercal H)\right) \\ &\Leftrightarrow \min \|S - H^\intercal H\|_F^2.\end{aligned} \tag{4.6}$$

If the restriction (4.5) is relaxed using $H \geq 0$, i.e. nonnegative H, we will arrive at our SymNMF formulation. □

Theorem 4.2. *When $S = D^{-1/2}S^{\mathcal{G}}D^{-1/2}$,* (4.1) *is a relaxation of minimizing the normalized cut, as defined in* (1.2).

Proof. Let $H = (h_{ij})_{k\times n}$, where

$$h_{ij} = \frac{\sqrt{d_j}}{\sqrt{\operatorname{within}(P_i) + \operatorname{out}(P_i)}}[v_j \in P_i]. \tag{4.7}$$

Then we have

$$\begin{aligned}
\operatorname{tr}(HD^{-1/2}SD^{-1/2}H^\intercal) &= \sum_{\substack{1\le i\le k\\ 1\le j,l\le n}} \frac{h_{ij}w_{jl}h_{il}}{\sqrt{d_j d_l}} \\
&= \sum_{\substack{1\le i\le k\\ 1\le j,l\le n}} \frac{w_{jl}}{\operatorname{within}(P_i)+\operatorname{out}(P_i)}[v_j\in P_i][v_l\in P_i] \\
&= \sum_{i=1}^{k} \frac{\operatorname{within}(P_i)}{\operatorname{within}(P_i)+\operatorname{out}(P_i)} \\
&= k - \operatorname{ncut}(P_1,\dots,P_k),
\end{aligned}$$

and

$$\begin{aligned}
&(HH^\intercal)_{ij} \\
&= \sum_k h_{ik}h_{jk} \\
&= \sum_k \frac{d_k}{\sqrt{\operatorname{within}(P_i)+\operatorname{out}(P_i)}\sqrt{\operatorname{within}(P_j)+\operatorname{out}(P_j)}}[v_k\in P_i\cap P_j] \\
&= \frac{[i=j]}{\operatorname{within}(P_i)+\operatorname{out}(P_i)}\sum_k d_k[v_k\in P_i] \\
&= [i=j],
\end{aligned}$$

which means $HH^\intercal = I$. Similar to (4.6), we have

$$\begin{aligned}
\min \operatorname{ncut}(P_1,\dots,P_k) &\Leftrightarrow \min\{k-\operatorname{tr}(HD^{-1/2}SD^{-1/2}H^\intercal)\} \\
&\Leftrightarrow \max \operatorname{tr}(HD^{-1/2}SD^{-1/2}H^\intercal) \\
&\Leftrightarrow \min \|D^{-1/2}SD^{-1/2}-H^\intercal H\|_F^2.
\end{aligned} \tag{4.8}$$

When the restriction (4.7) is relaxed to $H \ge 0$, our SymNMF formulation is obtained. □

SymNMF is an effective algorithm for graph clustering [11], but for large k, improvements in computational efficiency are necessary. In the next section, we will demonstrated

Algorithm 4 Divide-and-Conquer Framework for Divisive Hierarchical Clustering

1: **Initialization**: One cluster containing all nodes.
2: **repeat**
3: Choose one of the clusters to split.
4: Split the chosen cluster into two clusters.
5: **until** there are k clusters (or other stopping criteria)

our improved algorithm—HierSymNMF2.

4.2 Hierarchical SymNMF for Large Scale Community Detection

The algorithm we introduce in this chapter uses a similar divide-and-conquer idea as in Chapter 3, as summarized in Algorithm 4, where a cluster is a community, and the task of splitting a community is performed by our rank-2 version of SymNMF. The decision to choose the next node to split is based on a criteria discussed in the next section. In the following sections, we denote S as the similarity matrix representing a graph $\mathcal{G}$, and S_c as the matrix representation of a community, i.e., a subgraph of $\mathcal{G}$ (the corresponding submatrix of S).

4.2.1 Splitting a Community Using Rank-2 SymNMF

Splitting a community is achieved by rank-2 SymNMF of $S_c \approx H^\intercal H$ where $H \in \mathbb{R}_+^{2\times n}$. The result H naturally induces a binary split of the community: suppose $H = (h_{ij})$, then

$$c_i = \begin{cases} 1, & h_{1i} > h_{2i}; \\ 0, & \text{otherwise.} \end{cases}$$

where c_i is the community assignment of the ith graph node.

A formal formulation of rank-2 SymNMF is the following optimization problem:

$$\min_{H\geq 0} \|S - H^\intercal H\|_F^2, \tag{4.9}$$

where $H \in \mathbb{R}_+^{2\times n}$. This is a special case of SymNMF when $k = 2$, which can be solved by a general SymNMF algorithm [57, 11]. However, by combining the *alternating nonnegative least squares* (ANLS) algorithm for SymNMF from [11] and the fast algorithm for rank-2 NMF from [10] (restated in Chapter 2), we can obtain a fast algorithm for rank-2 SymNMF.

First, we rewrite (4.9) into asymmetric form plus a penalty term [48]:

$$\min_{W,H\geq 0} \|S - W^\intercal H\|_F^2 + \alpha\|W - H\|_F^2, \tag{4.10}$$

where W and $H \in \mathbb{R}_+^{2\times n}$ and $\alpha > 0$ is a scalar parameter for the tradeoff between the approximation error and the difference between W and H. Formulation (4.10) can be solved using a two-block coordinate descent framework, alternating between the optimization for W and H. When we solve for W, (4.10) can be reformulated as

$$\min_{W^\intercal\geq 0} \left\| \begin{bmatrix} H^\intercal \\ \sqrt{\alpha} I_2 \end{bmatrix} W - \begin{bmatrix} S \\ \sqrt{\alpha} H \end{bmatrix} \right\|_F^2, \tag{4.11}$$

where I_2 is the 2×2 identity matrix. Similarly, when we solve for H, (4.10) can be reformulated as

$$\min_{H\geq 0} \left\| \begin{bmatrix} W^\intercal \\ \sqrt{\alpha} I_2 \end{bmatrix} H - \begin{bmatrix} S \\ \sqrt{\alpha} W \end{bmatrix} \right\|_F^2. \tag{4.12}$$

We note that both (4.11) and (4.12) are in the form of (2.6) with $k = 2$, which can be efficiently solved by Algorithm 2.

4.2.2 Choosing a Node to Split Based on Normalized Cut

The "best" community to split further is chosen by computing and comparing *splitting scores* for all current communities corresponding to the leaf nodes in the hierarchy. The proposed splitting scores are based on normalized cut. We make this choice because: 1) normalized cut determines whether a split is structurally effective since it measures the difference between intra- and inter- connections among network nodes; 2) for SymNMF, when S is the normalized adjacency matrix, the SymNMF objective function is equivalent to (a relaxation of) minimizing the normalized cut (Theorem 4.2), which is the preferred choice in graph clustering [11].

We illustrate our splitting criteria using an example graph shown in Figure 4.1. It originally has three communities P_1, P_2 and P_3, and the corresponding normalized cut is

$$\begin{aligned}\mathrm{ncut}(P_1, P_2, P_3) = &\frac{\mathrm{out}(P_1)}{\mathrm{within}(P_1) + \mathrm{out}(P_1)} + \frac{\mathrm{out}(P_2)}{\mathrm{within}(P_2) + \mathrm{out}(P_2)}\\ &+ \frac{\mathrm{out}(P_3)}{\mathrm{within}(P_3) + \mathrm{out}(P_3)}.\end{aligned}$$

The community P_3 is now split into two smaller communities Q_1 and Q_2 and normalized

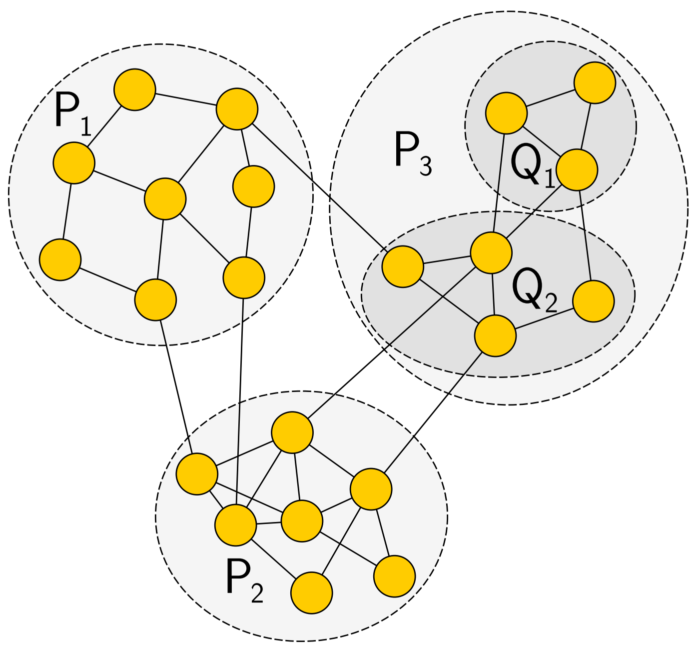


Figure 4.1: A graph for illustrating the splitting criteria for HierSymNMF2. The structure of the graph is inspired by Figure 1 from [60].

cut can be used to measure the goodness of this split. We consider three possibilities: (1)

Isolate P_3 and compute normalized cut of the split as

$$\text{ncut}\big|_{P_3}(Q_1, Q_2) = \frac{\text{out}\big|_{P_3}(Q_1)}{\text{within}(Q_1) + \text{out}\big|_{P_3}(Q_1)} + \frac{\text{out}\big|_{P_3}(Q_2)}{\text{within}(Q_2) + \text{out}\big|_{P_3}(Q_2)},$$

where the subscript P_3 means only consider the edges inside P_3. We denote the above criterion by `ncut_local`. (2) A more global criterion is to also consider the edges that go across P_3:

$$\text{ncut}(Q_1, Q_2) = \frac{\text{out}(Q_1)}{\text{within}(Q_1) + \text{out}(Q_1)} + \frac{\text{out}(Q_2)}{\text{within}(Q_2) + \text{out}(Q_2)}.$$

This criterion is denoted by `ncut_global`. (3) Minimize the global normalized cut using a greedy strategy. Specifically, choose the split that results in the minimal increase in the global normalized cut:

$$\begin{aligned}
&\text{ncut}(P_1, P_2, Q_1, Q_2) - \text{ncut}(P_1, P_2, P_3) \\
&= \frac{\text{out}(Q_1)}{\text{within}(Q_1) + \text{out}(Q_1)} + \frac{\text{out}(Q_2)}{\text{within}(Q_2) + \text{out}(Q_2)} - \frac{\text{out}(P_3)}{\text{within}(P_3) + \text{out}(P_3)}.
\end{aligned}$$

We denote this criterion by `ncut_global_diff` and will compare the performance of these three criteria in later sections.

4.3 Related Work

The study of network community detection dates back to the well known Kernighan-Lin algorithm from the early 1970s [61]. At that time, the network community detection problem was often formulated as a graph partitioning problem, which aims at "dividing the vertices" into a predefined number of non-overlapping "groups of predefined size, such that the number of edges lying between the groups is minimal" [62]. Many methods that produce good quality solutions were proposed but they were based on combinatorial optimization algorithms and were not scalable. Later, when it was discovered that graph partitioning is an important problem for balanced distribution of work loads in parallel computing, computer scientists developed many algorithms, such as METIS [63], SCOTCH [64] and Chaco [65], for graph partitioning of parallel communication problems. These algorithms usually follow a multilevel strategy, where a large graph is first coarsened to a smaller graph by recursively contracting multiple vertices into one vertex, and then the small graph is partitioned applying a method such as the Kernighan-Lin algorithm, and finally the partition is mapped back to the original graph with some refinement. Most of these algorithms (e.g. all three we mentioned above) scale well to very large networks containing millions of nodes.

A spectral clustering method that minimizes normalized cut was proposed as an image segmentation algorithm [66] and it soon became popular in the area of graph clustering. However, due to the time-consuming eigenvector/singular vector computation in this algorithm, it is not scalable to the case when the number of communities is large. The Graclus algorithm [67] by Dhillon, Guan and Kulis solved this issue by utilizing the mathematical equivalence between general cut or association objectives (including normalized cut and ratio association) and the weighted kernel k-means objective [68] and applying a multilevel framework. Kuang, Ding and Park discovered that the SymNMF (Symmetric Nonnegative Matrix Factorization) objective function is also equivalent to normalized cut and ratio association objective functions with a different relaxation from that in spectral clustering [57, 11]. This algorithm has better interpretability like many other NMF-based methods.

Girvan and Newman [60] produced pioneering work developing graph partitioning/clustering methods from a community detection viewpoint, which finds "groups of vertices which probably share common properties and/or play similar roles within the graph" [62]. Around that time period, many new algorithms were invented. Later, Newman and Girvan [69] proposed the modularity measurement for community detection, on which the biggest family of community detection algorithms is based [70]. A scalable example in this family of algorithms is the Louvain algorithm [71]. Several algorithms such as Walktrap [72] and Infomap [73] are based on random walk on graphs, with the idea that in a random walk, the probability of staying inside a community is higher than going to another community. The paper [62] provides a comprehensive review of the algorithms that appeared up until 2010.

The early overlapping community detection algorithms [74, 75] were not effective on large graphs. Lancichinetti et al. [76] proposed a scalable overlapping community detection algorithm—*Order Statistics Local Optimization Method* (OSLOM), which was based on a measurement similar to modularity but could handle overlapping communities. Yang and Leskovec studied properties of large scale overlapping communities [18] and proposed the BigClam algorithm [77]. They provided some large scale data sets with ground truth communities available to researchers, which have become standard test data sets. The BigClam algorithm seeks to fit a probabilistic generative model that satisfies certain community properties discovered in their studies [77, 78]. [79] proposed another overlapping community detection algorithm called NISE, based on seed set expansion, which starts with a seed set generated by Graclus or other methods and uses random walk to obtain overlapping communities. These algorithms that are dedicated to community detection have demonstrated better performance in terms of discovering the ground truth communities compared to the traditional graph partitioning algorithms.

Recently, [70] proposed a new non-overlapping community detection algorithm, *Scalable Community Detection* (SCD). Network communities of good quality should have stronger intra-connections than inter-connections. Previous algorithms measure such strength of connectivity only through the number of edges. The uniqueness of SCD is that it is based on a triangular structure of edges. The goal is to identify communities where each node forms more triangles with nodes inside the community than with nodes outside the community.

On the other hand, nonnegative matrix factorization (NMF)-based methods exhibit superior interpretability in many application areas such as text mining [10, 1] and image analysis [12, 2]. In this chapter, we will show that our NMF-based algorithm has competitive performance and scalability for community detection. Although our algorithm currently only handles non-overlapping community detection, it achieves comparable or even better quality than the state-of-the-art overlapping community detection algorithms (such as BigClam) in our extensive tests. Our algorithm is inspired by SymNMF [57, 11] for graph clustering and HierNMF2 [10] for fast document clustering.

4.4 Experiments

4.4.1 Methods for Comparison

We compare our algorithm with some recent algorithms mentioned in Section 4.3. We use 8 threads for all methods that support multi-threading. For NISE we are only able to use one thread because its parallel version exits with error in our experiments. For all the algorithms, default parameters are used if not specified. To better communicate the results, below are the labels that denote each algorithm, which will be used in the following tables:

- `h2-n(g)(d)-a(x)`: These labels represent several versions of our algorithm. Here `h2` stands for HierSymNMF2, `n` for the `ncut_local` criterion, `ng` for the `ncut_global` criterion, and `ngd` for the `ncut_global_diff` criterion (see previous sections for the definitions of these criteria); '`a`' means that we compute the real normalized cut using the original **a**jacency matrix; and '`x`' indicates that an appro**x**imated normalized cut is computed using the normalized adjacency matrix, which usually results in faster computations. We stop our algorithm after $k-1$ binary splits where k is the number of communities to find. Theoretically, this will generate k communities. However, we remove fully disconnected communities, as outliers since they are often far from significant because of their unusually small sizes and they correspond to all-zero submatrices in the graph adjacency matrix, which does not have a meaningful rank-2 representation. Therefore, the final number of communities are usually slightly smaller than k, as will be shown in the "Experiment Results" section.

- `SCD`: SCD algorithm [70].

- `BigClam`: BigClam algorithm [77].

Table 4.1: Some statistics for ground truth communities from SNAP.

Data set	#Nodes	#Edges	Nodes that belong to 0 Community		Nodes that belong to 1 Community		
			Count	%	Count	%	Rel %
DBLP06	317080	1049866	56082	17.69%	150192	47.37%	57.55%
Youtube	1134890	2987624	1082215	95.36%	32613	2.87%	61.91%
Amazon	334863	925872	14915	4.45%	10604	3.17%	3.31%
LiveJournal	3997962	34681189	2850014	71.29%	394234	9.86%	34.34%
Friendster	65608366	1806067135	57663417	87.89%	3546017	5.40%	44.63%
Orkut	3072441	117185083	750142	24.42%	128094	4.17%	5.52%

The last few columns show the number of nodes that do not belong to any communities and the number of nodes that belong to only one community. The "Rel %" is the number of nodes that belong to one community divided by the number of nodes that belong to at least one community.

- `Graclus`: Graclus algorithm [67].
- `NISE`: An improved version of NISE that is published in 2016 [80].

4.4.2 Data sets

The data used for the experimental results of this chapter are mostly from SNAP data sets [81, 18]. In our study, we found that the ground-truth information in SNAP is incomplete, for example, a large percentage of nodes does not belong to any ground-truth community. Table 4.1 shows some statistics regarding the number of communities to which each node belongs. Although all of these data sets can be conveniently accessed on the SNAP website as a graph with ground-truth communities, **DBLP06** is the only data set with a complete raw data set openly available to the public. The other five data sets (**Youtube**, **Amazon**, **LiveJournal**, **Friendster** and **Orkut**) were obtained by crawling the web, and they are far from being complete. Crawling large complex graphs is challenging by itself that may need extensive and specialized research efforts. We do not aim to solve this issue in this work. The **Orkut** and **Youtube** data sets can be acquired from [82]. Detailed descriptions are available explaining the crawling procedure and analysis of the completeness. It has been concluded that the **Orkut** and **Youtube** data sets are not complete. Such incompleteness in crawled data sets is expected due to intrinsic restrictions of web crawling such as rate limit and privacy protection. The **Friendster** data was crawled by the ArchiveTeam and the **LiveJournal** data comes from [83]. The **Amazon** data was crawled by the SNAP group [84]. However, information on how the data were collected and processed, and analysis of data completeness are not available.

Possible reasons that many nodes in these data sets do not belong to any communities are: (1) SNAP removed communities with less than three nodes, which caused some nodes to "lose" their memberships; (2) The well known incompleteness of crawled data sets; (3) For social networks (**Youtube**, **LiveJournal**, **Friendster**, and **Orkut**), it is common that a user does not join any user groups; (4) SNAP used the data set from [83] to generate the **DBLP06** data set, which was published in 2006. At that time, the DBLP database was not as mature and complete as it is today. Another issue of the above data sets is that all nodes are anonymized, which ensures protection of user privacy, but limits our ability to interpret community detection results.

The DBLP data is openly accessible, and is provided using a highly structured format—XML. We reconstructed the co-authorship network and ground-truth communities from a recent DBLP snapshot to obtain a more recent and complete DBLP data set with all of the meta information preserved (see the following subsection). Although the other data sets which we currently cannot improve are also valuable, our goal is to obtain new information from comparison of community detection results and ground truth communities, rather than simply recovering the ground truth communities.

*Constructing the **DBLP15** Data Set*

DBLP is an online reference for bibliographic information on major computer science publications. [85]. As of June 17, 2015, DBLP has indexed 4,316 conferences, 1,417 journals and 1,573,969 authors [86]. The whole DBLP data set is provided in a well formatted XML file. The snapshot/release version of the data we use can be accessed at `http://dblp.dagstuhl.de/xml/release/dblp-2015-06-02.xml.gz`.

The structure of this XML file is illustrated in Figure 4.2. The root element is the `dblp`

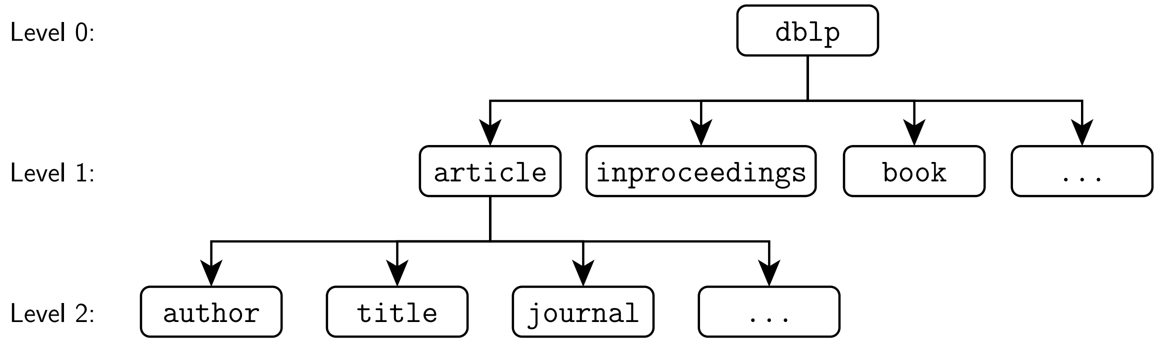


Figure 4.2: Structure of `dblp.xml`.

element. We call the children of the root elements *Level 1 elements* and the children of Level 1 elements *Level 2 elements*, and so on. Level 1 elements represent the individual data records [87], such as `article` and `book`, etc. Since publication-venue relation makes more sense for the journal and conference papers, and these two types of publications occupy most of DBLP, we consider only `article` and `inproceedings` elements when constructing our data set. Level 2 elements contain the meta-information about the publications, such as title, authors, journal/proceeding names, etc.

Our goal is to obtain a co-authorship network and ground truth information (venue-author relation) from the XML file. Although the XML file is highly structured, such a task is still not straightforward due to the ambiguity of entities, such as conflicts or changes of author names, various abbreviations, or even journal name change. DBLP resolves the author ambiguity issue by using a unique number for each author. However, the venue ambiguity is still an issue in DBLP: there are no unique identifiers for venues. Fortunately, each record in DBLP has a unique key and most papers' keys contain the venue information as follows:

$$\underbrace{\texttt{journals}}_{\text{venue type}}/\underbrace{\texttt{siamsc}}_{\text{venue identifier}}/\overbrace{\texttt{KimP11}}^{\text{publication identifier}}$$

However, there are still a few exceptions. To examine the validity of venue identifiers efficiently, we manually examine the identifiers not listed in the journal and conference index provided by the DBLP website, since such indices seem to be maintained by humans and assumed to be reliable. Using this process we found 5240 unique venues (journals or conferences).

Now unique identifiers for both authors and venues make extracting the network and community information very reasonable. The next step is to create a node for each author, and create a link between two authors if they have ever coauthored in the same publication. For community information, each venue is a community, and an author belongs to a community if he/she has published in the corresponding venue.

A few authors do not have any coauthor in the DBLP database, and become isolated nodes in the generated network. Thus, we remove these authors. However, after removing those authors, some venues/communities become empty because all of their authors are removed. So we remove those empty communities. After this cleaning, we obtained 1,509,944 authors in 5,147 communities (venues).

This cleaned network has 51,328 (weakly) connected components, where the largest connected component contains 1,357,781 nodes, which takes 89.9% of all nodes. The remaining 51,327 connected components are all small, the largest of which has only 37 nodes. We take the largest connected component as the network to study. By extracting the largest connected component, we obtain a network with 1,357,781 nodes, 6,369,212 edges and 5,146 ground truth communities. The ground truth communities were divided into connected components, obtaining 93,824 communities. The divided ground truth communities were used for comparison with detected communities.

The new **DBLP15** data set is available at `https://github.com/smallk/smallk_data/tree/master/dblp_ground_truth`.

4.4.3 Experiment Results

We run our experiments on a server with two Intel E5-2620 processors, each having six cores, and 377 GB memory. The results are listed in Tables 4.2 to 4.9.

Table 4.2: Community detection results on DBLP06: internal measures

Algorithm	Number of Clusters	Coverage	Algorithm Time (s)	Total Time (s)	Average Ncut
`h2-n-a`	4982	98.57%	612.99	614.12	0.2089
`h2-n-x`	4981	98.55%	587.98	589.10	0.2174
`h2-ng-a`	4984	98.48%	921.99	923.14	0.1922
`h2-ng-x`	4982	98.50%	872.48	873.64	0.1921
`h2-ngd-a`	4986	98.64%	882.27	883.41	0.1767
`h2-ngd-x`	4984	98.66%	908.31	909.46	0.1774
`SCD`	139986	100.00%	1.89	4.52	0.8091
`BigClam`	5000	90.57%	N/A	230.59	0.6083
`Graclus`	5000	100.00%	161.70	162.01	0.2228
`NISE`	5463	99.33%	501.38	501.53	0.2026

Table 4.3: Community detection results on DBLP06: external measures

Algorithm	Number of Clusters	F1	Precision	Recall	Reverse Precision	Reverse Recall
`h2-n-a`	3312	0.4355	0.8804	0.5242	0.9005	0.4030
`h2-n-x`	3298	0.4236	0.8855	0.5071	0.9007	0.3937
`h2-ng-a`	3211	0.4417	0.8708	0.5492	0.8490	0.3996
`h2-ng-x`	3118	0.4374	0.8742	0.5497	0.8574	0.3898
`h2-ngd-a`	3192	0.4577	0.8575	0.5800	0.8719	0.4091
`h2-ngd-x`	3138	0.4534	0.8541	0.5808	0.8768	0.4008
`SCD`	34705	0.4644	0.9817	0.1268	0.7053	0.9755
`BigClam`	4952	0.3778	0.4857	0.6807	0.9269	0.3121
`Graclus`	4633	0.4765	0.6915	0.6006	0.8852	0.4517
`NISE`	4903	0.4118	0.5735	0.7942	0.9518	0.3552

In the "internal measures" table, "coverage" measures the percentage of nodes which are assigned to at least one community; "algorithm time" and "total time" provide the runtime information. We list two measures of runtime since our algorithm (and also NISE) implemented in MATLAB directly uses a processed matrix in memory as its input. Other

Table 4.4: Community detection results on Amazon: internal measures

Algorithm	Number of Clusters	Coverage	Algorithm Time (s)	Total Time (s)	Average Ncut
h2-n-a	4989	98.84%	466.99	468.09	0.1657
h2-n-x	4988	98.80%	452.05	453.13	0.1711
h2-ng-a	4990	98.73%	537.82	538.91	0.1617
h2-ng-x	4988	98.66%	514.71	515.81	0.1709
h2-ngd-a	4990	98.82%	573.64	574.73	0.1491
h2-ngd-x	4990	98.79%	560.86	561.96	0.1545
SCD	141405	100.00%	1.86	4.37	0.8418
BigClam	5000	97.31%	N/A	169.51	0.3198
Graclus	5000	100.00%	119.25	119.45	0.1450
NISE	5182	99.63%	990.84	990.86	0.1118

Table 4.5: Community detection results on Amazon: external measures

Algorithm	Number of Clusters	F1	Precision	Recall	Reverse Precision	Reverse Recall
h2-n-a	1069	0.7883	0.9747	0.8179	0.9057	0.7593
h2-n-x	1038	0.7717	0.9787	0.8109	0.9070	0.7311
h2-ng-a	1209	0.7422	0.9657	0.7247	0.8748	0.7622
h2-ng-x	1185	0.7268	0.9655	0.7152	0.8743	0.7372
h2-ngd-a	1181	0.7813	0.9698	0.7741	0.8867	0.7922
h2-ngd-x	1168	0.7725	0.9702	0.7681	0.8869	0.7792
SCD	3841	0.6202	0.9998	0.3166	0.8186	0.9948
BigClam	1447	0.8389	0.9718	0.7824	0.9574	0.8744
Graclus	991	0.8555	0.9356	0.9471	0.9892	0.7525
NISE	2612	0.6673	0.6666	0.9733	0.9807	0.5390

Table 4.6: Community detection results on Youtube: internal measures

Algorithm	Number of Clusters	Coverage	Algorithm Time (s)	Total Time (s)	Average Ncut
`h2-n-a`	3782	98.10%	1182.39	1185.94	0.1681
`h2-n-x`	3780	98.01%	1189.09	1192.66	0.1634
`h2-ng-a`	3798	98.00%	1885.15	1888.71	0.1520
`h2-ng-x`	3851	98.14%	1816.98	1820.45	0.1491
`h2-ngd-a`	3886	98.27%	1613.13	1616.57	0.1395
`h2-ngd-x`	3874	98.22%	1621.04	1624.50	0.1428
`SCD`	998722	100.00%	12.03	20.39	0.9882
`BigClam`	5000	41.51%	N/A	2379.84	0.7398
`Graclus`	5000	100.00%	2160.11	2168.36	0.4919
`NISE`	5162	99.96%	2598.25	2598.66	0.4313

Table 4.7: Community detection results on Youtube: external measures

Algorithm	Number of Clusters	F1	Precision	Recall	Reverse Precision	Reverse Recall
`h2-n-a`	189	0.2907	0.9639	0.5247	0.9810	0.0403
`h2-n-x`	193	0.2972	0.9645	0.5411	0.9790	0.0412
`h2-ng-a`	241	0.2935	0.8684	0.5969	0.9315	0.0516
`h2-ng-x`	259	0.3027	0.8932	0.5915	0.9467	0.0551
`h2-ngd-a`	227	0.3030	0.9299	0.5694	0.9594	0.0484
`h2-ngd-x`	238	0.2978	0.9394	0.5476	0.9633	0.0507
`SCD`	27864	0.3652	0.9709	0.1330	0.4453	0.9841
`BigClam`	3850	0.2354	0.3755	0.5187	0.4743	0.2370
`Graclus`	3802	0.3827	0.5761	0.5348	0.6532	0.4148
`NISE`	3778	0.2720	0.4762	0.7180	0.9912	0.2580

Table 4.8: Community detection results on DBLP15: internal measures

Algorithm	Number of Clusters	Coverage	Algorithm Time (s)	Total Time (s)	Average Ncut
h2-n-a	4982	99.66%	1648.73	1654.71	0.1702
h2-n-x	4982	99.67%	1666.13	1672.01	0.1743
h2-ng-a	4984	99.62%	3262.76	3268.63	0.1606
h2-ng-x	4984	99.64%	3220.70	3226.57	0.1568
h2-ngd-a	4987	99.69%	2558.80	2564.60	0.1457
h2-ngd-x	4987	99.70%	2503.58	2509.38	0.1463
SCD	565235	100.00%	16.89	33.22	0.8357
BigClam	5000	65.07%	N/A	1352.57	0.6761
Graclus	5000	100.00%	1980.38	1987.97	0.2732
NISE	5101	86.77%	945.15	945.90	0.3482

Table 4.9: Community detection results on DBLP15: external measures

Algorithm	Number of Clusters	F1	Precision	Recall	Reverse Precision	Reverse Recall
h2-n-a	4982	0.3028	0.7282	0.7000	0.9830	0.0445
h2-n-x	4982	0.2994	0.7229	0.6986	0.9833	0.0442
h2-ng-a	4984	0.3025	0.7188	0.7164	0.9066	0.0440
h2-ng-x	4984	0.2992	0.6978	0.7275	0.9095	0.0439
h2-ngd-a	4987	0.3036	0.6963	0.7455	0.9640	0.0446
h2-ngd-x	4987	0.3016	0.6839	0.7512	0.9658	0.0446
SCD	565235	0.3477	0.8684	0.1050	0.5803	0.8218
BigClam	5000	0.0784	0.2357	0.9875	0.6806	0.0192
Graclus	5000	0.0861	0.2411	0.9874	0.7576	0.0275
NISE	5101	0.0955	0.3606	0.8307	0.7066	0.0253

algorithms must first read the graph stored as an edge list or an adjacency list and convert the graph to the appropriate internal representation. Therefore, we use "algorithm time" to measure the algorithm runtime without the time for reading and converting the graph, which is reported by the algorithms themselves. The "total time" is the wall clock time for running the algorithm, including the time for reading and converting the graph, which is measured with an external timer. `BigClam` reports its algorithm time as the sum of time used in each core and therefore the results are not comparable. For completeness, we added the data loading and preprocessing time, which is measured separately, to obtain a "total time" for MATLAB algorithms (our algorithm and NISE).

The "number of clusters" in the "internal measures" table is different across different methods due to the following reasons. The `SCD` algorithm does not provide an interface for specifying the number of communities to detect, and instead detects the number of communities automatically. For other algorithms, we specify the number of communities to detect as 5000. The actual number of communities generated by HierSymNMF2 is usually smaller than 5000, as discussed in the "Methods for Comparison" section. Also, the number of communities generated by `NISE` are usually a little larger than 5000, which is also an expected behavior [80].

In the "external measures" table, the "reverse precision" and "reverse recall" refer to the scores computed as if the ground truth communities are treated as detected communities and the detected communities are treated as the ground truth, respectively. Note that the number of clusters in "external measures" is smaller than the one in "internal measures" due to the removal of nodes that do not appear in the ground truth.

We have the following observations from the experimental results: (1) Our HierSymNMF2 algorithm has significant advantages over other methods in average normalized cut on most data sets except the **Amazon** data set. On the **Amazon** data set, HierSymNMF2 achieves much lower average normalized cut than `SCD` and `BigClam`, and the variant `h2-ngd-a` obtained comparable average normalized cut (0.1491) versus `Graclus`

(0.1450), which is not as good as NISE (0.1118) though. (2) HierSymNMF2 runs slower than most other algorithms on **DBLP06** and **DBLP15**. On the **Youtube** data set, HierSymNMF2 runs faster than `BigClam`, `Graclus` and `NISE`. On the **Amazon** data set, HierSymNMF2 runs faster than `NISE`, but slower than other methods. (3) HierSymNMF2 achieves better F1 score than `BigClam` and `NISE` on all the data sets we used. `Graclus` has better F1 score than HierSymNMF2 on **DBLP06**, **Amazon**, **Youtube** data sets but obtained an unusually low F1 score on the **DBLP15** data set. `SCD` achieves higher F1 scores than HierSymNMF2. However, `SCD` often discovers a significantly larger number of (non-overlapping) communities than expected and has very unbalanced precision and recall scores compared to other algorithms. The `SCD` algorithm finds a number of communities as it finds the communities and the number of communities cannot be given to `SCD` as an input. The `SCD` algorithm starts by assigning an initial partitioning of the graph heuristically. In short, in the initial partitioning, each node and all its neighbors form a community, and special care is taken to ensure that no node belongs to more than one community. As a result, this initial step often creates a lot more number of communities than the optimal number, though later refining procedures may reduce the number of communities. As can be seen from the experiment results, when compared to `BigClam`, `Graclus`, `NISE` and our proposed algorithms that take the number of communities as an input, a much larger number of communities that the `SCD` generates does not necessarily translate to a better overall community detection result in terms of either normalized cut or F1 scores.

CHAPTER 5

HYBRID CLUSTERING USING JOINTNMF

5.1 Motivation

As we can see in previous chapters, (DC-)NMF and (Hier)SymNMF(2) have good performance on text clustering and graph clustering, respectively. There are also numerous data sets containing both text content and connection structure. For example, in a data set of research papers or patents, papers or patents have text content where the citations or co-author relationships define the connection structure; in a data set of emails, email messages have text content and the sender-recipient relations define a hypergraph structure where one email may have multiple recipients. When the connection structure is represented as edges in a graph, in the former case the text content is associated with graph nodes while in the latter case the text content is associated with hypergraph edges. In this chapter, a hybrid clustering method is designed to utilize both content and connection structure information, thus taking advantage of the full information provided in the data.

Since NMF for content clustering and SymNMF for graph clustering have the same underlying matrix factorization framework, they can be merged at the objective function level, becoming the JointNMF we will discuss for the rest of the chapter.

5.2 Hybrid Clustering via JointNMF

First we assume that the text content is associated with the graph nodes. For example, a collection of research papers or patents can be represneted in a graph where the content information of each paper or patent is a graph node and the citation information provides the graph connection information. As in previous chapters, we assume that a data set's text information is represented in a nonnegative matrix $X \in \mathbb{R}_+^{m\times n}$ and the graph structure is represented in a nonnegative symmetric matrix $S \in \mathbb{R}_+^{n\times n}$, where m is the number of features, and n is the number of data items.

The hybrid clustering method we propose finds a low rank representation that simultaneously represents the text content and the graph structure of the data items by jointly optimizing the combined NMF (1.1) and SymNMF (4.1) objective functions:

$$\min_{W\geq 0, H\geq 0} \alpha_1 ||X - WH||_F^2 + \alpha_2 ||S - H^\intercal H||_F^2, \tag{5.1}$$

where $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$ are the weighting parameters. By adjusting the parameters α_i, we can emphasize one over the other. In the extreme case, some α_i can be set to zero: e.g. when $\alpha_2 = 0$ in the above, we are only concerned with the content, when $\alpha_1 = 0$, we only pay attention to the structural information and ignore the content. Excluding these special cases, we can assume $\alpha_1 = 1$ without loss of generality and Equation (5.1) becomes

$$\min_{W\geq 0, H\geq 0} ||X - WH||_F^2 + \alpha ||S - H^\intercal H||_F^2, \tag{5.2}$$

with $\alpha \geq 0$ as the weighting parameter.

Now we extend our method to hypergraphs where the text content is associated with hypergraph nodes. Once this is done, it would be natural to extend our method further to the cases where text is associated with graph or hypergraph edges due to the duality that exists between edges and nodes of a hypergraph and the fact that a graph can be treated as a special case of a hypergraph.

A hypergraph $\mathcal{H}$ is a pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is the set of vertices and $\mathcal{E} = \{e_1, \ldots, e_p : \ e_i \subset \mathcal{V}\}$ is the set of hyperedges. Unlike a graph edge, a hypergraph edge e_i may connect more than two vertices in the graph. Such a hypergraph $\mathcal{H}$ can be represented by an incidence matrix $B = (b_{ij}) \in \mathbb{R}^{n\times p}$, where

$$b_{ij} = \begin{cases} 1, & v_i \in e_j; \\ 0, & \text{otherwise.} \end{cases}$$

The dual hypergraph $\mathcal{H}^*$ is the hypergraph corresponding to the incidence matrix $B^\intercal$.

Assume there's a k-way partition of the vertices $(\mathcal{V}_1, \dots, \mathcal{V}_k)$ where $\mathcal{V}_1 \cup \cdots \cup \mathcal{V}_k = \mathcal{V}$ and $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ for all $1 \le i \ne j \le k$. Define the matrix $H = (h_{ij}) \in \mathbb{R}^{k\times n}$ as

$$h_{ij} = \frac{[v_j \in \mathcal{V}_i]}{\sqrt{d_v(j)} \left(\sum_{v_l \in \mathcal{V}_i} \frac{1}{d_v(l)} \right)^{1/2}}, \tag{5.3}$$

which is a normalized partition indicator matrix where

$$[v_j \in \mathcal{V}_i] = \begin{cases} 1, & v_j \in \mathcal{V}_i, \\ 0, & \text{otherwise,} \end{cases}$$

and $d_v(l) = \sum_{j=1}^p b_{lj}$ is the degree of vertex v_l. It is shown in [88] that the following optimization problem

$$\max_H \operatorname{tr} HSH^\intercal \tag{5.4}$$

is equivalent to minimizing the hypergraph normalized cut as defined in [88], where

$$S = D_v^{-1/2} B D_e^{-1} B^\intercal D_v^{-1/2} \tag{5.5}$$

is symmetric,

$$D_v = \operatorname{diag}(d_v(1), \dots, d_v(n)), \quad D_e = \operatorname{diag}(d_e(1), \dots, d_e(p)),$$

and $d_e(l) = \sum_{i=1}^{n} b_{il}$ is the degree of edge e_l. Following the same argument as in [57], it can be shown that (5.4) is equivalent to $\min_H \|S - H^\intercal H\|_F^2$ and by relaxing constraint (5.3) to $H \geq 0$, we obtain the objective function of SymNMF. Therefore, in the case of a hypergraph, we can use the matrix S defined in Equation (5.5) as the similarity matrix in Equation (5.2).

There are many ways to find a solution for the objective function (5.2). Theoretically, a Newton-like algorithm can be developed to directly solve (5.2). However, as pointed out in [11], a Newton-like algorithm can not utilize the sparsity of X and S for speeding up because the matrices $X - WH$ and $S - H^\intercal H$ need to be computed explicitly and thus the sparsity will be destroyed. On the other hand, an alternating nonnegative least square (ANLS) algorithm can be sped up with sparsity. To apply an ANLS-like algorithm that can utilize the sparse nature of text documents and associated networks, we propose reformulating (5.2) in the following form with a penalty term

$$\min_{W,H,\tilde{H}\geq 0} ||X - WH||_F^2 + \alpha||S - \tilde{H}^\intercal H||_F^2 + \beta\|\tilde{H} - H\|_F^2, \tag{5.6}$$

where $\tilde{H} \in \mathbb{R}_+^{k\times n}$ and $\beta \geq 0$ is the regularization parameter. This reformulation is motivated from our earlier work to generate an algorithm that is based on the block coordinate descent (BCD) scheme so that each sub-problem in the BCD is a nonnegativity constrained least squares (NLS) problem for which we have developed a highly efficient algorithm and optimized open-source software [5]. Then Equation (5.6) can be solved using a 3-block coordinate descent (BCD) scheme, i.e. minimize the objective function with respect to W,

$\tilde{H}$ and H in turn. Specifically, we solve the following three subproblems in turn:

$$\min_{W \geq 0} \|H^\intercal W^\intercal - X^\intercal\|_F, \tag{5.7}$$

$$\min_{\tilde{H} \geq 0} \left\| \begin{bmatrix} \sqrt{\alpha} H^\intercal \\ \sqrt{\beta} I_k \end{bmatrix} \tilde{H} - \begin{bmatrix} \sqrt{\alpha} S \\ \sqrt{\beta} H \end{bmatrix} \right\|_F, \tag{5.8}$$

$$\min_{H \geq 0} \left\| \begin{bmatrix} W \\ \sqrt{\alpha} \tilde{H}^\intercal \\ \sqrt{\beta} I_k \end{bmatrix} H - \begin{bmatrix} X \\ \sqrt{\alpha} S \\ \sqrt{\beta} \tilde{H} \end{bmatrix} \right\|_F, \tag{5.9}$$

where each subproblem is simply a nonnegative least squares problem (NNLS). Thus, Algorithm 1 can be used to find the optimal solution in a finite number of operations and ensures that the solution is in the feasible region. The above three block BCD algorithm converges to a stationary point according to Bertsekas' theorem [35]. The identity submatrices I_k in the above equations make the problem better conditioned than the subproblems in the standard NMF that uses two block BCD alternating updating W and H. We solve each NLS problem using the block principal pivoting (BPP) algorithm [31]. Theoretically, to force H to be identical to $\tilde{H}$, the value of the parameter β has to be infinity. This problem has been studied extensively and we use a scheme similar to that proposed in [89]. It should be pointed out that in [31] it is shown that algorithms based on the BCD framework have guaranteed convergence to a stationary point, whereas, popular and easy to implement algorithms such as Multiplicative Updating (MU) may not converge. In addition, extensive experiments show that the BPP method is faster and more accurate than MU.

5.3 Related Work

In addition to content, attribute, which usually refers to categorical information, can also be associated with the graph edges or nodes. For example, in Figure 5.1, "Age" and "City" are node attributes, "friend" and "work" are edge attributes. Content is usually more arbitrary,

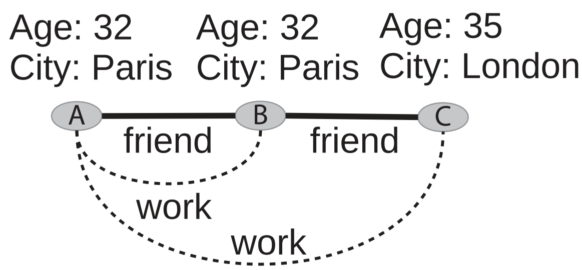


Figure 5.1: A graph with edge attribute and node attribute. The graph has three people as nodes. An edge labeled “friend” connects two people that are friends and another edge labeled “work” connects two people that work together. This figure is from [90] with slight modification.

more unstructured and of larger size, such as text documents or pictures.

Some literature may not distinguish these two terms and refer to text documents as attributes. We explicitly make this distinction since: 1) Some algorithms that handle categorical attributes cannot be easily extended to arbitrary documents 2) Edge attributes and edge content are usually treated in very different ways, as we will see in later sections.

We organize our discussions in two categories

- text clustering augmented by network structure information (graph clustering), which can also be viewed as graph clustering augmented by information on the node. Depending on whether the information on the node is content or attributes, problem formulation and algorithms may differ.
- graph clustering augmented by information on the edge

5.3.1 Graph Clustering augmented with Node Attributes/Content

Most of the available hybrid clustering methods involve graph clustering/community detection augmented with node attributes/content information. We list the algorithms that we have investigated in Table 5.1. The column **A/C** indicates whether the algorithms are designed for node attribute or node content. Although it is possible to encode arbitrary content as categorical attributes, with a large number of attributes, an algorithm designed for node attributes may not extend well to arbitrary content because the categorical encoding may not

Table 5.1: Graph clustering methods utilizing node attributes/content. The first seven methods above the horizontal line can be viewed as topic modeling augmented with network information.

Name	**Ref.**	**A/C**†	**D/U**‡	**Type**
PHITS-PLSA*	[91]	content	D	generative
Link-LDA*	[92]	content	D	generative
Pairwise Link-LDA	[93]	content	D/U	generative
Link-PLSA-LDA	[93]	content	D	generative
Topic-Link LDA	[94]	content	U	generative
LTHM	[95]	content	D	generative
RTM	[96]	content	U(D)	generative
BAGC	[97]	attribute	U(D)	generative
GBAGC	[98]	attribute	U(D)	generative
CESNA	[99]	attribute	U	generative
PCL-DC	[100]	content	D	discriminative
NetPLSA	[101]	content	U	topic model + network regularization
GenClus	[102]	both	D	topic model + network regularization
NetScan	[103]	both	U	combinatorial optimization
SA-Cluster	[104]	attribute	U	augmented graph, distance based
Inc-Cluster	[105]	attribute	U	augmented graph, distance based
CODICIL	[106]	both	U	augmented graph
2JointMF/mJointMF	[107]	both	D/U	matrix factorization
Entropy based*	[108]	both	U	joint optimize entrophy + modularity
HGPA	[109]	both	U	ensemble
CSPA	[109]	both	U	ensemble
Selection*	[110]	both	U	selection

* These methods were not given names in the original papers. The names given here are based on either 1) how they are referred to in the literature or 2) the characteristics of the method

† A/C stands for Attribute/Content.

‡ D/U refers to Directed/Undirected graphs.

capture the content well and the algorithm may not scale to large numbers of attributes. On the other hand, an algorithm designed for text content may not work well for attributes since the quality of such an algorithm may rely on the amount of content, but attributes are usually short and categorical. In Table 5.1, the column **D/U** indicates whether the algorithm is designed for directed or undirected graphs. Some authors claim that algorithms designed for undirected graphs can easily be extended to the directed case. Theses algorithms are labeled with "U(D)". However, such claims depend on the specific assumptions in the problem formulation and the underlying model/algorithm. For example, in generative models, all methods that are designed for undirected graphs generate the links based on some similarity between nodes. The column **Type** summarizes the basic idea behind each method.

We see that almost half of the methods in Table 5.1 use generative models. In general, a generative clustering model learns a latent cluster indicator (for each node), based on which all the content and links are generated, usually controlled by other parameters. Such latent cluster indicator could be a vector that measures how likely a node belongs to each cluster (for soft clustering), or a single variable that assigns a node to a specific cluster (hard clustering). Such a latent cluster indicator could be generated by other parameters or it could be a parameter itself. Table 5.2 summarizes this information.

Although these algorithms generate documents/attributes and links with various assumptions and rules, many share similar strategies. For generating documents/attributes, all of the LDA or PLSA-based models use a generative topic model while other algorithms generate them from the latent cluster indicator using some other rules. For generating links, BAGC, GBAGC, and CESNA generate links based on similarity of latent cluster indicator, RTM generates links based on similarity of generated documents, Pairwise Link-LDA generates links based on similarity of topics generated from a latent cluster indicator, PHITS-PLSA and Link-LDA cite documents based on a latent cluster indicator, and finally LTHM cites documents that have similar topics.

Link-PLSA-LDA assumes that the citation graph of the documents are bipartite, which

Table 5.2: Generative graph clustering methods utilizing nodes attributes/content.

Name	**LCI* generated?**	**LCI* form**	**Methods†**
PHITS-PLSA	No	Vector	DG
Link-LDA	Yes	Vector	LDA, DG
Pairwise Link-LDA	Yes	Vector	LDA, DG
Link-PLSA-LDA	Yes	Vector	LDA, BP
Topic-Link LDA	Yes	Vector	LDA
LTHM	Yes	Vector	LDA, DF
RTM	Yes	Vector	LDA, DF
BAGC	Yes	Variable	DG
GBAGC	Yes	Variable	DG
CESNA	No	Vector	DG

* LCI: Latent Cluster Indicator

† Methods refer to the method(s) used to generate the documents. LDA: Latent Dirichlet allocation. DF generates the documents, i.e. every word in the document, first and then generates the links based on the documents. DG generates documents/attributes and the graph can be generated simultaneously from the latent cluster indicator. BP (for Link-PLSA-LDA) means that the model assumes the directed graph is bipartite.

means "each document can either be cited or be a citing document, but not both" [93]. It first generates cited documents based on PLSA, and then generates citing documents (and links) based on cited documents from Link-LDA. Topic-Link LDA detects document clusters and author communities jointly. Therefore, a link is generated based on both the latent cluster indicators of topics and the latent cluster indicators of authors. Yang, Jin, Chi, and Zhu [100] argued that generative models failed to consider additional factors that could affect the community memberships and isolate the content that is irrelevant to community memberships. They proposed a discriminative algorithm PCL-DC, which overcomes these two shortcomings. NetPLSA and GenClus also uses generative topic models. But their focus is on maximizing the log-likelihood as the objective function, instead of the generative process. They add network consistency as a regularization term to the objective function, which tries to enforce the rule that connected documents must have similar distribution over topics. NetScan forms the hybrid clustering problem as a combinatorial optimization problem — connected k-center problem, which tries to cluster the node content/attributes while keeping each cluster of nodes connected in the graph. The authors proved that such problems are NP-complete and provided a heuristic algorithm. The Inc-Cluster algorithm and SA-Cluster algorithm are based on the same idea. However, Inc-Cluster is more efficient than SA-Cluster. These two algorithms first construct an augmented graph by adding "attribute vertices" and "attribute edges" to reflect node attributes. The augmented graph is then used to compute the random walk distance as a distance measure of vertices in the original graph. Finally, they perform an adapted K-Medoids algorithm on the graph using the random walk distance, to obtain the clustering result. CODICIL also uses an augmented graph. It first adds edges between every two nodes that have very similar content to get an augmented graph. Then it samples the most relevant edges from the augmented graph, and then outputs this simplified graph into another graph partitioning algorithm such as METIS.

Matrix factorization-based methods usually find a low dimensional representation of the data. Such a low dimensional representation can be either given to other vector based clustering methods such as K-means (e.g. spectral clustering using SVD) or directly interpreted as cluster indicator (e.g. NMF). When there is more than one data source (e.g. documents and graph), matrix factorization based methods can be extended to factor several matrices jointly. The formulation of 2JointMF is

$$\min_{W_0, W_1, H} \|A - W_0 H\|_F^2 + \|C - W_1 H\|_F^2 + \text{regularization terms}, \tag{5.10}$$

with constraints $H \geq 0$ and $0 \leq W_0 H \leq 1$, where A is the adjacency matrix of the graph (can be asymmetric) and C encodes the documents/attributes. The regularization terms are

$$\lambda \sum_{j=1}^{n} \|H(:, j)\|_1^2 + \eta \left(\|W_0\|_F^2 + \|W_1\|_F^2\right).$$

In this formulation, one can directly read clustering information from the matrix H. The method mJointMF adds more joint factorization terms when there are more data sources.

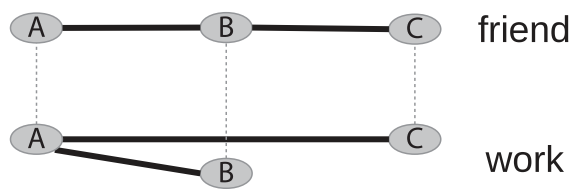


Figure 5.2: Graph with edge attributes viewed as a multi-layer graph. Figure 5.1 is the original edge attributed graph. This figure is taken from [90] with slight modifications.

To solve (5.10), the authors used a 3-block coordinate descent framework to alternatively minimize the objective function w.r.t. W_0, W_1 and H where each subproblem is either a least squares problem or can be decomposed into several linear constrained least squares problems with one right hand side.

The rest of the methods are summarized in Table 5.1. The entropy-based method jointly minimizes the entropy of document clusters and maximizes modularity of graph clusters. The HGPA and CSPA methods assemble the result of a graph clustering algorithm and the result of a document clustering algorithm into a combined result. The selection method argues that some graphs have clear structure content, but some do not. It selects a graph clustering algorithm when the graph has clear structure information and selects attributes only algorithms when the graph has ambiguous structure information.

5.3.2 Text Clustering Augmented with Network Structure Information

While node attributes and node content are usually encoded similarly, edge attributes and edge content are treated very differently. In this section, we treat graph clustering with edge attributes/content as equivalent to text clustering with network structure information. Graphs with edge attributes are usually treated as a multi-layer graph. For example, the graph in Figure 5.1 can be viewed in a multi-layer way, as illustrated in Figure 5.2 (with node attributes removed). The survey [90] provides a good review for the methods that utilize edge attribute information.

Edge content are unstructured and accordingly cannot be treated in the same way as the attributes. Qi, Aggarwal and Huang [111] proposed matrix factorization methods EIMF-

Lap and EIMF-LP, which detects communities in a graph with edge content. Instead of clustering nodes directly, they start with partitioning the edges. They find a low dimensional vector representation of each edge where both the graph structure and edge content are reflected. Suppose the graph is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\mathcal{E} = \{e_1, \ldots, e_m\}$. They encode the graph with an edge-vertex matrix instead of an adjacency matrix. In an edge-vertex matrix Γ, $\Gamma_{i,j} = 1$ if and only if the vertex v_j is an endpoint of the edge e_i (otherwise it is zero). To find vector representations of edges that reflect graph structure, they perform the following matrix factorization:

$$\min_{E,V} \|E^\intercal V - \Gamma\|_F^2,$$

where columns of $E \in \mathbb{R}^{k\times m}$ are vector representations of edges and columns of $V \in \mathbb{R}^{k\times n}$ are vector representations of vertices. Then they represent a vertex as an arithmetic average of the vector representations of all the edges incident to that vertex. In other words, we have $V = E\Delta$, where

$$\Delta_{i,j} = \begin{cases} 1/\deg(v_j) & \text{vertex } v_j \text{ is an endpoint of edge } e_i \\ 0 & \text{otherwise} \end{cases}$$

Then the objective function of EIMF (edge induced matrix factorization) is

$$\min_{E} \|E^\intercal E\Delta - \Gamma\|_F^2.$$

The authors provide two methods to encode edges to reflect edge content. Suppose the edge content is already encoded by m d-dimensional vectors stored in a $d \times m$ matrix C. The first method is through Laplacian eigenmaps, which embeds d-dimensional vectors into a k dimensional space by minimizing $\mathrm{tr}(E^\intercal LE)$, where L is the normalized graph Laplacian $L = D^{-1/2}(D - S)D^{-1/2}$. The matrix S is the cosine similarity matrix constructed from matrix C and $D = \mathrm{diag}(s_1, \ldots, s_m)$, where s_i is the sum of elements in the ith row vector of matrix S. When combined with EIMF, the formulation becomes

$$\min_{E} \|E^\intercal E\Delta - \Gamma\|_F^2 + \lambda \cdot \mathrm{tr}(E^\intercal LE). \tag{5.11}$$

This method is called EIMF-Lap. To avoid tuning the parameter λ, the authors provide an alternative method through linear projection. This method makes a stronger assumption that there exists an $k \times d$ matrix W, such that $E = WC$. When combined with EIMF, the formulation becomes:

$$\min_{W} \|C^\intercal W^\intercal WC\Delta - \Gamma\|_F^2. \tag{5.12}$$

This method is called EIMF-LP.

The vector representation of edges E are given to a clustering algorithm such as K-means, and clustering of edges into edge communities is obtained. Each edge community $\mathcal{C}_i$ can then induce a vertex community $\mathcal{P}_i$ where all the endpoints of the edges in $\mathcal{C}_i$ are contained. The authors chose the assumptions ($V = E\Delta$ and $E = WC$) such that the optimization problem (5.11) and (5.12) are all convex. Then the authors proposed multiplicative update algorithms to solve both formulations.

5.3.3 Differences with Other Joint Matrix Factorization Methods

Besides the 2JointMF [107] mentioned above, the use of joint matrix factorization for clustering can also be seen in [112, 113], all of which consider clustering using information from different sources. Our JointNMF is different from 2JointMF in the following ways: (1) JointNMF has nonnegative constraints on all matrix factors while 2JointMF has nonnegative constraint on H only. The nonnegative constraints on all factors usually lead to better interpretations of the result. For example, the W factor in our formulation can be interpreted as topic vectors due to its nonnegativity. (2) 2JointMF factors the graph matrix in a way similar to factoring a general feature matrix, while the symmetric factorization from

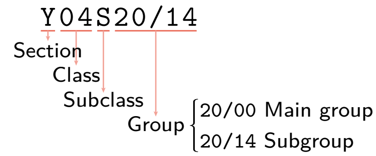


Figure 5.3: An example classification label in the CPC scheme

JointNMF acknowledges the symmetric similarity relation encoded in a graph and also relates itself with minimizing normalized cut. (3) The constraint $0 \leq W_0 H \leq 1$ makes 2JointMF computationally much more difficult [114, 115]. The algorithm in [112] also jointly minimizes several NMF objectives. However, they do not consider graph information and therefore SymNMF was not in their formulation. Although the objective function in [113] looks very similar to what we propose in the chapter, the matrices in the formulas have different meanings and their formulation is used only in the context of graph clustering.

5.4 Clustering US Patent, BlogCatalog and Flickr Data

All experiments were performed on a server with two Intel(R) Xeon(R) CPU E5-2680 v3 CPUs and 377GB memory.

The main data set used for the experiments is the US patent claim and citation data from PatentsView[1]. Some advantages of using US patents as a data source are: (1) the openness, centralized management and availability of relatively structured data format makes the patent data easier to obtain and process; (2) the abundance of the patent database ensures enough samples that can be studied; (3) patents were carefully assigned with classification labels, and such labels were examined by patent examiners; therefore the classification information can be used as a relatively reliable ground truth.

We use the Cooperative Patent Classification (CPC) system, where each classification label has the scheme illustrated in Figure 5.3. We select 13 CPC classes (A22, A42, B06, B09, B68, C06, C13, C14, C40, D02, D10, F22, Y04) and use patents under each class to

[1] http://www.patentsview.org

Table 5.3: Some statistics of US patent data sets.

Class	#Patents	#Citations	#Groups
A22	4976	28746	230
A42	4213	29285	134
B06	2938	11549	82
B09	3522	17302	38
B68	790	2433	93
C06	3347	17562	141
C13	1010	3717	87
C14	583	1125	69
C40	3748	28854	41
D02	3170	11216	158
D10	2548	8486	154
F22	3040	7977	359
Y04	3242	21518	76

construct 13 different data sets[2]. For each data set, we first construct the term-document matrix representing the patent claims and the graph adjacency matrix representing the patent citation relations. Our algorithm requires a symmetric adjacency matrix and therefore we treat the citation graph as undirected by ignoring the directions. We then clean the data by removing terms that appear very infrequently and documents that are too short or duplicated, and extract the largest connected components of the graph. Finally, we apply tf-idf to the term-document matrix, normalize its columns to have unit 2-norm, obtaining the matrix X, and let S be $D^{-1/2}AD^{-1/2}$, where $A \in \mathbb{R}^{n\times n}$ is the adjacency matrix, $D = \mathrm{diag}(d_1, \ldots, d_n)$ and $d_i = \sum_{j=1}^{n} A_{ij}$ is the degree of vertex i. We use CPC groups as ground truth clusters. Some statistics about these data sets (after cleaning) are listed in Table 5.3.

To verify our algorithm on other types of data, we also use the BlogCatalog data set from [116] and the Flickr data set from [117]. These data sets have users as graph nodes and represent user commenting and friendship relations as graph edges. The content comes from user generated keywords/tags that are used to describe their blog articles (BlogCatalog) or photos (Flickr), which is different from traditional text content. The ground truth clusters

[2]These data sets are available at `http://smallk.github.io/pages_about.html`

Table 5.4: Some statistics of BlogCatalog and Flickr data sets.

Data	#Nodes	#Edges	#Tags	#Groud truth clusters
BlogCatalog	31228	782584	5387	60
Flickr	32576	2749800	77234	170

of BlogCatalog data set are defined by categories of each blog and the ones for the Flickr data set are defined by user groups. We apply the same preprocessing as for the US patent data sets. Some statistics regarding these two data sets (after preprocessing) are listed in Table 5.4.

Since the ground truth clusters have overlapping, we use average F1 score and Rand index, as discussed in Section 1.2.3 , as the measures for the evaluation of the clustering results.

We compare our algorithm with NMF and SymNMF, which have leading performance in text clustering and graph clustering, respectively. For hybrid clustering, we choose PCL-DC [100] to compare with based on its popularity and source code availability. While our method is based on nonnegative matrix factorization, PCL-DC is a probabilistic method that combines a conditional model for link analysis and a discriminative model for content analysis. Although we mentioned many other algorithms in Section 5.3, we found that for other algorithms, either the code is not available or the code is available but we encountered runtime errors during experimental tests. Both JointNMF and PCL-DC have parameters to set. For JointNMF, we let the default parameter be $\alpha = \|X\|_F^2/\|S\|_F^2$, meaning half-half balance between graph clustering and text clustering, and set $\beta = \alpha\|S\|_{max}$, where $\|S\|_{max}$ is the maximum absolute value of elements in S. The authors of PCL-DC do not provide a method to specify its regularization parameter λ. Therefore, it is important to first study how the parameter change will affect the algorithm performance. It is found that for $\lambda < 1$, PCL-DC sometimes becomes extremely slow, such that it may take weeks to run over all the data sets (estimated based on sampling run). Therefore, λ is varied within $[1, 20]$. In Figure 5.4, we show how the average F1 score changes when λ varies in that range for the

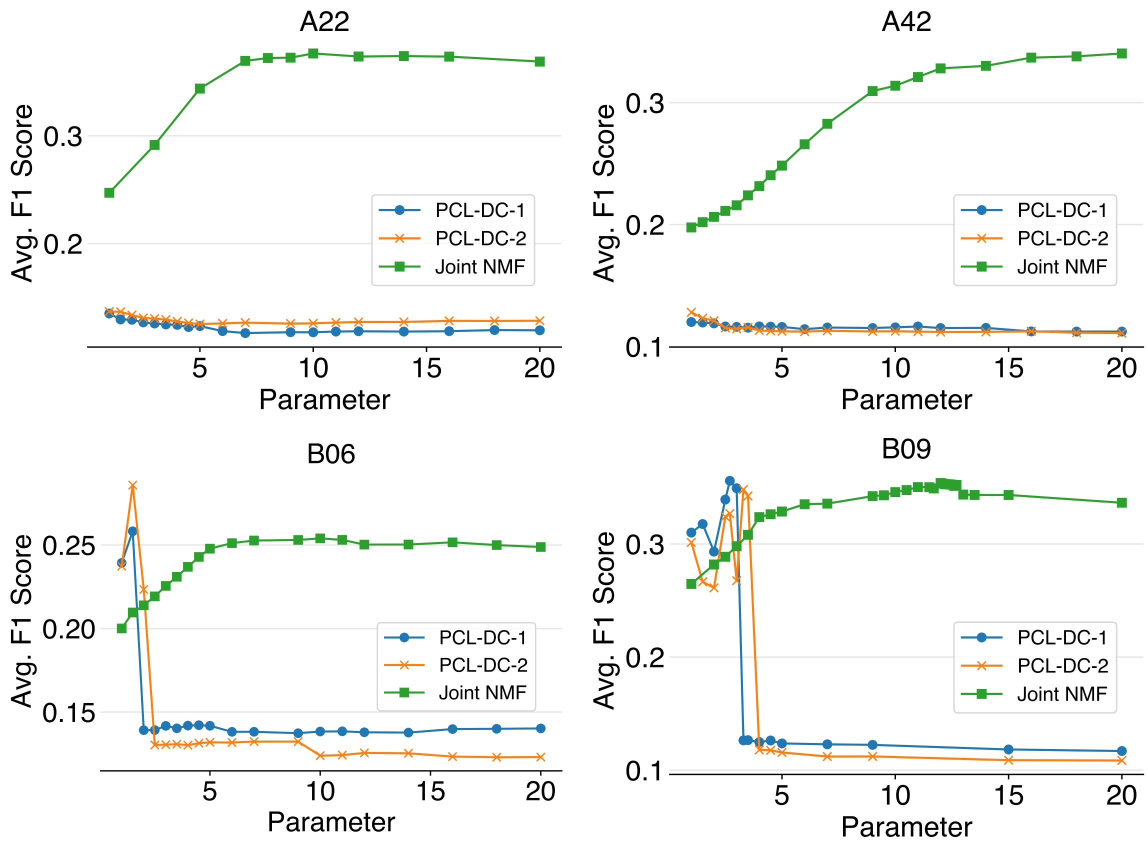


Figure 5.4: Parameter sensitivity of PCL-DC and JointNMF. The parameter of PCL-DC is λ and the parameter of JointNMF is α.

first four data sets listed in Table 5.3. The code of PCL-DC[3] provides two models (popularity link model and productivity link model), which we label as PCL-DC-1 and PCL-DC-2, respectively. The performance change of JointNMF when its parameter α varies in the same range is also studied. We observe that the PCL-DC is either worse than JointNMF or very sensitive to the parameters, and it is concluded that when λ exceeds a certain threshold (depending on the data), there is a large drop in clustering quality. Therefore, to have a tolerable run time while having a fair clustering quality, $\lambda = 1$ is chosen for the comparison experiments. The results of the comparison are listed in Table 5.5 to Table 5.7, where each value is the average over 10 runs.

Using these patent data sets, from our experiments it can be observed that: (1) JointNMF

[3] `https://homepage.cs.uiowa.edu/~tyng/codes/community_detection.zip`

Table 5.5: Hybrid clustering results: comparison of average F1 scores

Class	JointNMF	NMF	SymNMF	PCL-DC-1	PCL-DC-2
A22	**0.3730**	0.2293	0.3457	0.1351	0.1369
A42	**0.3215**	0.1779	0.3199	0.1201	0.1280
B06	**0.2502**	0.1905	0.2307	0.2393	0.2373
B09	**0.3336**	0.2449	0.2690	0.3101	0.3014
B68	0.3806	0.3044	0.3730	**0.4034**	0.3671
C06	**0.2257**	0.1830	0.2004	0.1156	0.1158
C13	**0.2990**	0.2664	0.2953	0.2616	0.2224
C14	0.3584	0.3232	**0.3603**	0.2692	0.2659
C40	0.1939	0.1709	0.1673	0.1951	**0.1981**
D02	**0.2990**	0.2131	0.2683	0.1756	0.2268
D10	**0.3046**	0.2452	0.2783	0.1612	0.2999
F22	**0.3006**	0.2211	0.2926	0.1533	0.1388
Y04	0.2489	0.2029	0.2019	**0.2599**	0.2596
blogcatalog	0.2038	0.2150	0.0750	**0.2754**	0.2754
flickr	0.1545	0.0748	**0.1660**	0.0855	0.0855

Table 5.6: Hybrid clustering results: comparison of rand index

Class	JointNMF	NMF	SymNMF	PCL-DC-1	PCL-DC-2
A22	**0.9785**	0.9768	0.9772	0.9274	0.9489
A42	**0.9650**	0.9633	0.9647	0.9225	0.9318
B06	**0.9368**	0.9357	0.9024	0.8775	0.8815
B09	**0.8497**	0.8387	0.7600	0.8464	0.8333
B68	0.9496	0.9423	**0.9508**	0.9272	0.8897
C06	0.9175	0.9150	**0.9182**	0.8969	0.8967
C13	0.8918	0.8873	**0.8927**	0.8598	0.8485
C14	**0.9086**	0.9036	0.9071	0.8233	0.7934
C40	0.6575	0.6507	**0.6820**	0.6593	0.6692
D02	**0.9612**	0.9594	0.9578	0.8922	0.8831
D10	**0.9080**	0.9048	0.9075	0.8676	0.8771
F22	0.9811	0.9797	**0.9816**	0.9554	0.9549
Y04	**0.8879**	0.8853	0.8697	0.8668	0.8622
blogcatalog	0.7572	**0.7652**	0.6173	0.7259	0.7259
flickr	0.0560	0.0409	**0.0782**	0.0620	0.0620

Table 5.7: Hybrid clustering results: comparison of run time (seconds)

Class	JointNMF	NMF	SymNMF	PCL-DC-1	PCL-DC-2
A22	769.4	304.4	219.2	**55.6**	57.5
A42	311.9	161.9	163.1	**24.3**	24.8
B06	193.8	115.8	**59.8**	444.5	1800.8
B09	145.6	109.6	**48.2**	406.6	588.8
B68	48.2	60.8	**7.6**	288.3	439.0
C06	489.8	269.0	160.6	21.1	**20.9**
C13	70.9	76.1	**8.8**	421.5	377.2
C14	29.5	25.0	**4.7**	220.6	83.8
C40	240.8	127.8	**54.3**	394.0	597.3
D02	534.5	238.5	**117.3**	1623.5	831.8
D10	280.8	155.4	95.9	**14.7**	1728.4
F22	1294.1	404.4	267.2	38.4	**36.7**
Y04	291.9	125.8	**103.8**	1568.3	987.6
blogcatalog	401.3	**222.8**	1515.6	4463.4	4522.4
flickr	12455.6	2437.9	3504.5	**1181.3**	1236.0

usually has the best average F1 scores, and its average F1 score is almost always better than that of NMF or SymNMF alone; (2) JointNMF and SymNMF have the best rand index; (3) SymNMF is usually the fasted algorithm; (4) The run time varies in a very different pattern between NMF based methods and PCL-DC. The algorithms for both NMF based methods and PCL-DC are iterative. For NMF based methods, the run time of each iteration is linear with respect to data size (e.g. number of nodes and edges) and cubic with respect to the number of clusters [10]. For PCL-DC, the run time of each iteration is linear with respect to both data size and the number of clusters [100]. If we compare Table 5.7 with Table 5.3, we can observe that for NMF methods the number of clusters does dominate the run time but for PCL-DC the run time is rather unpredictable, which may suggest that the convergence behavior of PCL-DC is not consistent over different data sets. On BlogCatalog and Flickr data sets, which have different kinds of content and graph edges, the performance varies depending on the data. However, the performance of JointNMF is comparable to the best method with the exception of run time on the Flickr data set.

In conclusion, for patent data sets, based on content and citations, JointNMF produces better quality solutions for clustering; for prediction of pairwise connection, both JointNMF and SymNMF perform well; speed-wise, JointNMF is not the fastest, but is comparable to other methods. On other types of data, the performance of each method varies, and JointNMF generates comparable results. The JointNMF method has other advantages: its parameter has explicit meanings (weight between text and graph), the clustering quality is not very sensitive to the parameter setting, and its default parameter works very well.

5.5 Other Applications

In this section we present additional applications of our JointNMF framework beyond clustering. We demonstrate our JointNMF on other potential applications such as citation recommendations of papers/patents and activity/leader detection in an organization.

5.5.1 Citation Recommendation

When applied to papers/patents with citations or web pages with hyperlinks, the formulation (5.2) can also be understood as finding a basis W for the text space, such that under this basis, the representation (coordinates) of the documents can also reflect their linkage information. Therefore, when we express a new vector $\boldsymbol{x}$ in the text space using the basis W, i.e. finding a vector $\boldsymbol{h}$ that solves the following optimization problem

$$\min_{\boldsymbol{h}\geq 0} \|\boldsymbol{x} - W\boldsymbol{h}\|_2, \tag{5.13}$$

we can use closeness of $\boldsymbol{h}$ to the column vectors in H to decide how likely the new document represented by $\boldsymbol{h}$ should cite some of the documents in H. For example, one can recommend a new document to cite the i-th original document if the i-th entry of $H^\intercal \boldsymbol{h}$ is larger than certain threshold. Since the matrix S is approximated by $H^\intercal H$, we can treat $H^\intercal \boldsymbol{h}$ as an augmented column of S, representing the relation between the original documents and the new document. Another method is to set the threshold for the cosine similarity between $\boldsymbol{h}$

and column vectors in H. It will be observed that each method has its advantages.

For this task, we use the paper title/abstract and citation data **cit-HepTh** from SNAP[81], which contains 27,770 papers from January 1993 to April 2003 in the hep-th (high energy physics - theory) section of arXiv. Note that this is a different task from clustering and therefore the data preprocessing procedure is a little different: the raw adjacency matrix for S (i.e. $S = A$) is used. The normalized version $D^{-1/2}AD^{-1/2}$ is related to minimizing the normalized cut [57] and therefore good for clustering. Here the raw adjacency matrix is a better indicator of citations, which is used as an input that the algorithm learns from, instead of a basis for clustering.

To evaluate our method, the data is separated into training and test sets by treating papers published earlier than 2003 as the training set and papers published in 2003 as the test set. We use JointNMF to learn a matrix W from the document and citation relations in the training set, and then make predictions of citations for documents in the test set and compare the predictions with the actual citations.

To verify that the W computed by our algorithm indeed reflects the network structure better, we also design several baseline methods. A naive method is to predict citations based on number of words shared by two documents. One method based on NMF is to learn the matrix W used in (5.13) only by NMF, i.e. $\min_{W\geq 0, H\geq 0} \|X_{train} - WH\|_F$. Another method based on NMF is to directly learn the $\boldsymbol{h}$ vector in (5.13) using $\min_{W,H,\boldsymbol{h}\geq 0} \|[X_{train}, \boldsymbol{x}] - W[H, \boldsymbol{h}]\|_F$. For the two NMF-based methods, the rest of the steps for making predictions are the same as JointNMF, once the matrix W or the vector $\boldsymbol{h}$ is obtained. In this subsection, we denote these two NMF based methods as NMF-1 and NMF-2, respectively.

For both prediction methods (compute $H^\intercal \boldsymbol{h}$, the inner product , or compute cosine similarity scores), a threshold is needed. Instead of evaluating these algorithms with a fixed threshold, we show the receiver operating characteristic (ROC) curve, which plots the true positive rate against the false positive rate at various threshold values. In general, the closer the curve is to the upper left corner of the graph, the better the algorithm results. Or

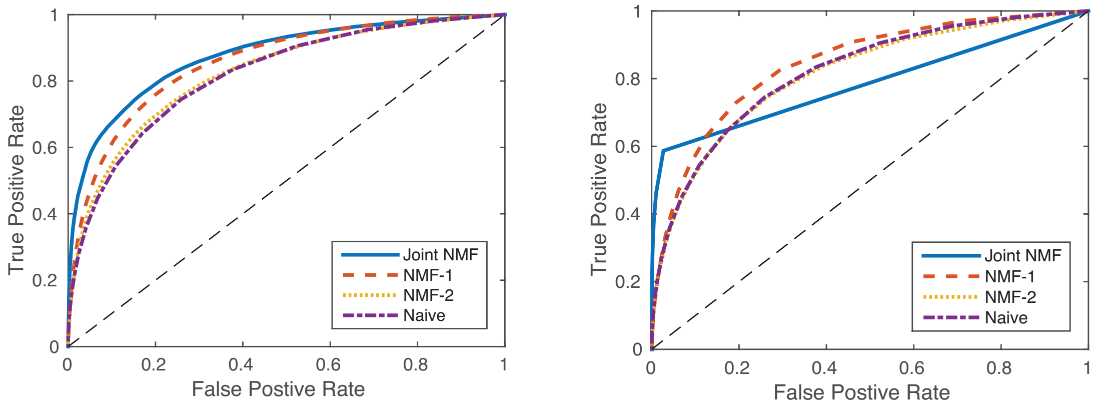


Figure 5.5: ROC curves for citation recommendation algorithms applied to paper abstract and citation data. The left uses cosine similarity for the prediction, while the right uses inner product.

quantitatively, the larger the area under the curve (AUC) is, the better.

Paper abstracts are used to extract text content. The experimental results are shown in Figure 5.5. Some observations are: when cosine similarity is used, JointNMF makes the overall best predictions, and when inner product is used, at certain threshold values JointNMF can achieve relatively high true positive rate with a very low false positive rate. One can choose which method to use based on requirements. A heuristic explanation of such a difference caused by using cosine similarity or inner product is as follows: The cosine similarity ignores the length of the similar content while the inner product does not. Thus, the low false positive rate on the right sub-figures of Figure 5.5 and 5.6 suggests that if two papers' abstracts/titles share a large amount of content (in the sense of bag-of-word model), it is very likely that one paper would cite the other.

The experiments are repeated using only paper titles as text content; similar results are observed, as shown in Figure 5.6. From the results we can observe that even with very little text information (such as paper titles), our method still works well.

5.5.2 Activity and Leader Detection from Enron Email Data

In an organization where various groups of people work on different subjects and engage in different activities, JointNMF can be used to detect such group structure, reveal the

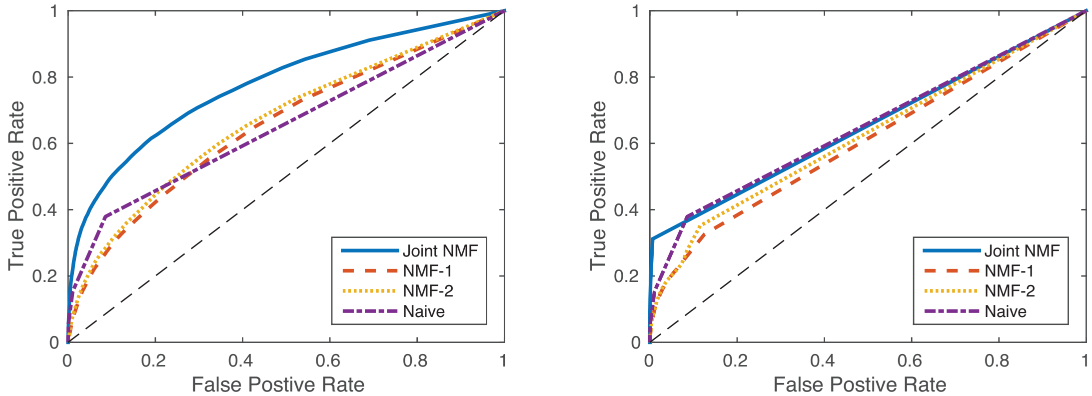


Figure 5.6: ROC curves for citation recommendation algorithms applied to paper title and citation data. The left uses cosine similarity for the prediction, while the right uses inner product.

working subject/activities and find administrators/leaders in the organization. We assume that (1) within-group communications (e.g. emails) reflects the subject on which the team is working/activities engaged in and (2) people involved in multiple groups would likely hold a higher position in the organization, since they may be in charge of these groups. Each communication can be seen as a hypergraph edge that connects all people involved in the communication and the communication content is the text associated with the edge. Clustering the text data can distinguish and identify different working subjects/activities and clustering the graph data can divide people into workgroups. JointNMF utilizes both types of data simultaneously and therefore can distinguish different groups of people working on the same subject and different subjects worked on by the same group of people. After clustering, one can count and compare the number of groups/clusters each person belongs to—the more groups a person belongs to, the more likely the person is in a leadership or administrative position.

A subset of Enron email data extracted by a group from UC Berkeley [4], containing 1702 emails is used. First we construct the term document matrix from email content and the hypergraph incidence matrix from email-sender/recipient relations. The hypergraph has Enron employees as vertices and their emails as edges, and a vertex is connected by an edge

[4] `http://bailando.sims.berkeley.edu/enron_email.html`

Table 5.8: Case study on Enron email data: frequency of number of memberships

#memberships	1	2	3	4	5	6	7	11
#employees	1069	149	45	17	8	7	1	1

if and only if the corresponding employee is the sender or a recipient of the corresponding email. After that, we clean the data by removing terms that appear very infrequently and emails that are too short or duplicated, and extracting the largest connected components of the hypergraph. The tf-idf transformation is then applied to the term-document matrix, whose columns are normalized to have unit 2-norm, which obtains the matrix X. S is computed using (5.5) in which B is the incidence matrix of the dual hypergraph. Finally, we apply JointNMF with $\alpha = \|X\|_F^2/\|S\|_F^2$ and $\beta = \alpha\|S\|_{max}$ to find 20 groups of employees. Note that since the dual hypergraph is used, the resulting clusters are clusters of emails rather than clusters of employees. To induce clusters of employees, one simply inserts employees involved in the same cluster of emails into one employee cluster. In this way, we can actually induce overlapping employee clusters from non-overlapping email clusters. It is assumed that an employee has j memberships if the employee belongs to j clusters. The number of memberships is counted for each employee and the frequency of each number is listed in Table 5.8. Employees that had at least 6 memberships are examined in online news and we found that they all held relatively high positions in Enron. Their names and positions are listed in Table 5.9. To see the effect of our algorithm on topic modeling, we list some topic keywords for each cluster in Table 5.10. It can be observed that some emails are communications about/with other companies and regulatory agencies (0,3,19); some are about administrative tasks or daily work (5,7,8,13,15,16,18); some are about legal issues (6,10); and some are related to the California energy crisis (2,11).

Table 5.9: Case study on Enron email data: employees that has j memberships ($j \geq 6$) and their positions in Enron

j	Name	Position in Enron
11	Steven Kean	Chief of staff
7	Jeff Dasovich	Governmental affairs executive
	Susan Mara	California director of Regulatory Affairs
	Richard Shapiro	VP of regulatory affairs
	Paul Kaufman	VP of Government Affairs
6	James Steffes	VP of Government Affairs
	Tim Belden	Head of trading
	Richard Sanders	VP of Enron Whole Sale Services
	Joe Hartsoe	VP of Federal Regulatory Affairs

VP: vice president

Table 5.10: Case study on Enron email data: topic keywords of clusters

#	Keywords
0	ubs, warburg, forecast, confidential, win
1	blackberry, handheld, wireless
2	california, power, confidential, tariff, pursuant
3	caiso, refund, ferc, proceedings
4	burrito, peace, things, price, market, board, california
5	document, fax, tonight, sign, back, attach, thanks
6	wholesale, policy, compliance, receipt, legal, service
7	enron, please, know, attach, meeting, contact, call, any, time
8	london, conference, meeting, next, week
9	handheld, blackberry, wireless, agreement, confidential
10	testify, witness, fault, burden, cut, budget
11	california, electricity, energy, price, market, power, rate, bill
12	recommendation, template, participant, management
13	passcode, please, effective, confidential, change
14	stanford, university, expert, try, best, mail, california
15	account, invoice, trust, fund, transfer
16	expense, report, employee, name , approve, amount
17	folder, info, audit, access, apollo, email, sensitivity, server
18	sent, talk, presentation, thanks, infrastructure, amendment
19	hpl, aep, agreement, compete, deal, arrangement

CHAPTER 6

RELATIONSHIPS AMONG VARIANTS OF NMF

6.1 Relation of NMF and SymNMF

Assuming $X \in \mathbb{R}_+^{m \times n}$, it is well known that the eigenvalue decomposition (EVD) of

$$M = \begin{bmatrix} & X \\ X^\intercal & \end{bmatrix} \tag{6.1}$$

has a corresponding relation with the SVD of X [41]. It is natural to ask whether there is a relation between SymNMF of M and NMF of X. Indeed, we have

Theorem 6.1. *Assume (W_*, H_*) is an optimal solution of (1.1), then $\tilde{H}_* = \begin{bmatrix} W_*^\intercal & H_* \end{bmatrix}$ is an optimal solution of*

$$\min_{\tilde{H} \geq 0} \left\| \begin{bmatrix} & E_{m,n} \\ E_{n,m} & \end{bmatrix} \circ \left(\begin{bmatrix} & X \\ X^\intercal & \end{bmatrix} - \tilde{H}^\intercal \tilde{H} \right) \right\|_F^2 \tag{6.2}$$

, where $\circ$ is the Hadamard product and $E_{m,n} \in \mathbb{R}^{m \times n}$ is a matrix whose elements are all one's.

Theorem 6.1, which can be easily verified by plugging in the form of $\tilde{H}_*$, shows that when we treat the zeros in matrix M as missing values (corresponding to unknown relations between documents and words) and formulate SymNMF in a matrix completion sense, such formulation is equivalent to NMF. Starting from the other direction, we have

Theorem 6.2. *Assume $\tilde{H}_*$ is an optimal solution of (4.1) and we split $\tilde{H}_*$ as $\tilde{H}_* = \begin{bmatrix} W_*^\intercal & H_* \end{bmatrix}$ where $W_* \in \mathbb{R}^{m \times k}$, $H_* \in \mathbb{R}^{k \times n}$ then (W_*, H_*) is an optimal solution of a* *regularized form of NMF*

$$\min_{W\geq 0, H\geq 0} \|X - WH\|_F^2 + \frac{1}{2}\|WW^\intercal\|_F^2 + \frac{1}{2}\|H^\intercal H\|_F^2. \tag{6.3}$$

The verification of Theorem 6.2 is also straightforward. However, unlike the SVD case, there is no obvious direct relation between NMF of X and SymNMF of M, without modification of their formulations. In particular, we have

Theorem 6.3. *Suppose $W_* \in \mathbb{R}^{m\times k}$, $H_* \in \mathbb{R}^{k\times n}$. If (W_*, H_*) satisfies the KKT condition of* (1.1) *and $\tilde{H}_* = \begin{bmatrix} W_*^\intercal & H_* \end{bmatrix}$ satisfies the KKT condition of* (4.1)*, then $W_* = 0$ and $H_* = 0$.*

Proof. The KKT conditions of (1.1) are

$$\begin{aligned} &W \geq 0, && H \geq 0, && (6.4)\\ &WHH^\intercal - XH^\intercal \geq 0, && H^\intercal W^\intercal W - X^\intercal W \geq 0, && (6.5)\\ &W \circ (WHH^\intercal - XH^\intercal) = 0, && H^\intercal \circ (H^\intercal W^\intercal W - X^\intercal W) = 0, && (6.6)\end{aligned}$$

and the KKT conditions of (4.1) are

$$\tilde{H} \geq 0, \tag{6.7}$$

$$\tilde{H}^\intercal \tilde{H} \tilde{H}^\intercal - M\tilde{H}^\intercal \geq 0, \tag{6.8}$$

$$\tilde{H}^\intercal \circ (\tilde{H}^\intercal \tilde{H} \tilde{H}^\intercal - M\tilde{H}^\intercal) = 0. \tag{6.9}$$

If we plug $\tilde{H}_* = \begin{bmatrix} W_*^\intercal & H_* \end{bmatrix}$ into (6.9), we get

$$W_* \circ (W_* W_*^\intercal W_* + W_* H_* H_*^\intercal - XH_*^\intercal) = 0, \tag{6.10}$$

$$H_*^\intercal \circ (H_*^\intercal W_*^\intercal W_* + H_*^\intercal H_* H_*^\intercal - X^\intercal W_*) = 0. \tag{6.11}$$

Next, we plug in (6.6) and obtain

$$W_* \circ (W_* W_*^\intercal W_*) = 0, \tag{6.12}$$

$$H_*^\intercal \circ (H_*^\intercal H_* H_*^\intercal) = 0. \tag{6.13}$$

Rewrite (6.12) in elementwise form

$$w_{ij} \sum_{p,q} w_{ip} w_{qp} w_{qj} = 0. \tag{6.14}$$

Sum (6.14) over i and j we obtain

$$\mathrm{tr}(W_*^\intercal W_* W_*^\intercal W_*) = 0, \tag{6.15}$$

which means $\|W_*^\intercal W_*\|_F = 0$ and therefore $W_* = 0$. Similarly, (6.13) means $H_* = 0$. $\square$

6.2 Relation of SymNMF and JointNMF

Now we assume that beyond the term-document matrix X we also have graph information of the documents which is encoded in a symmetric matrix S. Then the matrix M becomes

$$M = \begin{bmatrix} & X \\ X^\intercal & S \end{bmatrix}. \tag{6.16}$$

To analyze such data, we can apply SymNMF on M to get co-clusters of terms and documents. Or we can also use joint NMF for hybrid clustering:

$$\min_{W \geq 0, H \geq 0} \|X - WH\|_F^2 + \alpha \|S - H^\intercal H\|_F^2. \tag{6.17}$$

Similar to Theorem 6.1–Theorem 6.3, we have

Theorem 6.4. *Assume* (W_*, H_*) *is an optimal solution of* (6.17) *(with* $\alpha = 1/2$*), then*

$\tilde{H}_* = \begin{bmatrix} W_*^\intercal & H_* \end{bmatrix}$ *is an optimal solution of*

$$\min_{\tilde{H}\geq 0} \left\| \begin{bmatrix} & E_{m,n} \\ E_{n,m} & E_{n,n} \end{bmatrix} \circ \left(\begin{bmatrix} & X \\ X^\intercal & S \end{bmatrix} - \tilde{H}^\intercal \tilde{H} \right) \right\|_F^2 . \tag{6.18}$$

Theorem 6.5. *Assume $\tilde{H}_*$ is an optimal solution of* (4.1) *and we split $\tilde{H}_*$ as $\tilde{H}_* = \begin{bmatrix} W_*^\intercal & H_* \end{bmatrix}$ where $W_* \in \mathbb{R}^{m\times k}$, $H_* \in \mathbb{R}^{k\times n}$ then (W_*, H_*) is an optimal solution of a regularized form of joint NMF*

$$\min_{W\geq 0, H\geq 0} \|X - WH\|_F^2 + \frac{1}{2}\|S - H^\intercal H\|_F^2 + \frac{1}{2}\|WW^\intercal\|_F^2. \tag{6.19}$$

Theorem 6.6. *Suppose $W_* \in \mathbb{R}^{m\times k}$, $H_* \in \mathbb{R}^{k\times n}$. If (W_*, H_*) satisfies the KKT condition of* (6.17) *and $\tilde{H}_* = \begin{bmatrix} W_*^\intercal & H_* \end{bmatrix}$ satisfies the KKT condition of* (4.1)*, then $W_* = 0$.*

The proofs of Theorem 6.4–Theorem 6.6 have exactly the same idea and structure of the proofs of Theorem 6.1–Theorem 6.3, and are therefore omitted here.

CHAPTER 7
CONCLUSIONS

In this dissertation, we studied NMF variants for three important tasks for big data analysis: NMF and DC-NMF for text clustering/topic modeling, SymNMF and HierSymNMF2 for graph clustering/community detection and JointNMF for hybrid clustering. For each task, we studied existing literatures extensively, proposed proper NMF formulations, designed efficient algorithms and conducted extensive experiments. We also constructed some new data sets. We have seen that NMF based methods usually had better clustering quality and comparable computational cost with other state-of-the-art algorithms. Besides clustering, NMF based methods provided valuable insights of the data. The simple and unified framework of NMF based method has allowed an unified, convergent and efficient solution framework (BCD) and made it flexible to combine multiple sources of information. The good interpretability of NMF has made it possible to apply NMF based clustering algorithms to problems such as text based link prediction and leader/activity detection.

NMF has been applied in many real world projects. Using the methods described in this dissertation, the author has participated in many real world data analysis tasks such as summarizing people's opinion on sustainable technologies via text mining, detecting ceasefire violations in Yemen via analyzing Telegram messages, and identifying emerging, fading and evolution of technology via analysis of patents. The area of NMF is still developing. Some recent and ongoing work in which the author participated includes NMF based co-clustering and NMF based outlier detection. The author believes that more useful variants and applications of NMF will arise.

REFERENCES

[1] W. Xu, X. Liu, and Y. Gong, "Document Clustering Based on Non-negative Matrix Factorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, ser. SIGIR '03, New York, NY, USA: ACM, 2003, pp. 267–273.

[2] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[3] M. Hofree, J. P. Shen, H. Carter, A. Gross, and T. Ideker, "Network-based stratification of tumor mutations," *Nature Methods*, vol. 10, no. 11, pp. 1108–1115, Nov. 2013.

[4] A. Ozerov and C. Fvotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 3, pp. 550–563, 2010.

[5] B. Drake, S. Lee-Urban, and H. Park, *Smallk is a C++/Python high-performance software library for nonnegative matrix factorization (NMF) and hierarchical and flat clustering using the NMF; current version 1.6.2*, `http://smallk.github.io/`, 2017.

[6] R. Kannan, G. Ballard, and H. Park, "A high-performance parallel algorithm for non-negative matrix factorization," in *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '16, Barcelona, Spain: ACM, 2016, 9:1–9:11.

[7] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, ser. NIPS'00, Denver, CO: MIT Press, 2000, pp. 535–541.

[8] H. Kim and H. Park, "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 2007.

[9] J. Kim and H. Park, "Sparse nonnegative matrix factorization for clustering," Georgia Institute of Technology, Tech. Rep., 2008.

[10] D. Kuang and H. Park, "Fast Rank-2 Nonnegative Matrix Factorization for Hierarchical Document Clustering," in *Proceedings of the 19th ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, ser. KDD '13, New York, NY, USA: ACM, 2013, pp. 739–747.

[11] D. Kuang, S. Yun, and H. Park, "SymNMF: Nonnegative low-rank approximation of a similarity matrix for graph clustering," *Journal of Global Optimization*, vol. 62, no. 3, pp. 545–574, Nov. 2014.

[12] N. Gillis, D. Kuang, and H. Park, "Hierarchical Clustering of Hyperspectral Images Using Rank-Two Nonnegative Matrix Factorization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 4, pp. 2066–2078, Apr. 2015.

[13] L. Li, G. Lebanon, and H. Park, "Fast bregman divergence NMF using taylor expansion and coordinate descent," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12, Beijing, China: ACM, 2012, pp. 307–315.

[14] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[15] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, Award winning papers from the 19th International Conference on Pattern Recognition (ICPR)19th International Conference in Pattern Recognition (ICPR), vol. 31, no. 8, 651666, 2010.

[16] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, 395416, 2007.

[17] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, Aug. 2007.

[18] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, Oct. 2013.

[19] A. Sinclair and M. Jerrum, "Approximate counting, uniform generation and rapidly mixing Markov chains," *Information and Computation*, vol. 82, no. 1, pp. 93–133, Jul. 1989.

[20] L. M. Collins and C. W. Dent, "Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions," *Multivariate Behavioral Research*, vol. 23, no. 2, 231242, 1988.

[21] T. D. Klastorin, "The p-median problem for cluster analysis: A comparative test using the mixture model approach," *Management Science*, vol. 31, no. 1, 8495, 1985.

[22] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[23] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, no. Oct, 28372854, 2010.

[24] T. O. Kvalseth, "Entropy and Correlation: Some Comments," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, no. 3, pp. 517–519, May 1987.

[25] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, 2000, pp. 58–64.

[26] A. Strehl and J. Ghosh, "Cluster Ensembles — A Knowledge Reuse Framework for Combining Multiple Partitions," *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 583–617, 2002.

[27] ——, "Cluster ensembles-a knowledge reuse framework for combining partitionings," in *AAAI/IAAI*, 2002, pp. 93–99.

[28] L. N. F. Ana and A. K. Jain, "Robust data clustering," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*, vol. 2, Jun. 2003, II–128–II–133 vol.2.

[29] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, no. 3, p. 033 015, 2009.

[30] R. Du, D. Kuang, B. Drake, and H. Park, "DC-NMF: Nonnegative matrix factorization based on divide-and-conquer for fast clustering and topic modeling," *Journal of Global Optimization*, vol. 68, no. 4, pp. 777–798, 2017.

[31] J. Kim and H. Park, "Fast Nonnegative Matrix Factorization: An Active-Set-Like Method and Comparisons," *SIAM J. Sci. Comput.*, vol. 33, no. 6, pp. 3261–3281, Nov. 2011.

[32] R. Du, D. Kuang, B. Drake, and H. Park, "Hierarchical community detection via rank-2 symmetric nonnegative matrix factorization," *Computational Social Networks*, vol. 4, no. 1, p. 7, 2017.

[33] R. Du, B. Drake, and H. Park, "Hybrid clustering based on content and connection structure using joint nonnegative matrix factorization," *Journal of Global Optimization*, 2017.

[34] J. Kim, Y. He, and H. Park, "Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework," *Journal of Global Optimization*, vol. 58, no. 2, pp. 285–319, 2014.

[35] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[36] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear gausssseidel method under convex constraints," *Operations Research Letters*, vol. 26, no. 3, 127136, 2000.

[37] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, 211218, 1936.

[38] R. A. Horn and C. R. Johnson, Eds., *Matrix Analysis*. New York, NY, USA: Cambridge University Press, 1986.

[39] M. T. Chu and M. M. Lin, "Low-dimensional polytope approximation and its applications to nonnegative matrix factorization," *SIAM Journal on Scientific Computing*, vol. 30, no. 3, pp. 1131–1155, 2008. eprint: `http://dx.doi.org/10.1137/070680436`.

[40] J. E. Cohen and U. G. Rothblum, "Nonnegative ranks, decompositions, and factorizations of nonnegative matrices," *Linear Algebra and its Applications*, vol. 190, pp. 149 –168, 1993.

[41] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th. The Johns Hopkins University Press, 2013.

[42] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of Internet portals with machine learning," *Inf. Retr.*, vol. 3, no. 2, pp. 127–163, 2000.

[43] A. Globerson, G. Chechik, F. Pereira, and N. Tishby, "Euclidean embedding of co-occurrence data," *J. Mach. Learn. Res.*, vol. 8, pp. 2265–2295, 2007.

[44] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, 2004.

[45] C.-J. Hsieh and I. S. Dhillon, "Fast coordinate descent methods with variable selection for non-negative matrix factorization," in *17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '11)*, 2011, pp. 1064–1072.

[46] J. Kim and H. Park, "Toward faster nonnegative matrix factorization: A new algorithm and comparisons," in *ICDM '08: Proc. of the 8th IEEE Int. Conf. on Data Mining*, 2008, pp. 353–362.

[47] A. Cichocki and A. H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E92A, no. 3, pp. 708–721, 2009.

[48] N.-D. Ho, "Non-negative matrix factorization. algorithms and applications," PhD thesis, Universit catholique de Louvain, 2008.

[49] C.-J. Lin, "On the convergence of multiplicative update algorithms for nonnegative matrix factorization," *IEEE Trans. on Neural Networks*, vol. 18, no. 6, pp. 1589–1596, 2007.

[50] F. G. Nicolas Gillis, "Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization," *Neural Computation*, vol. 24, no. 4, pp. 1085–1105, 2012.

[51] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating non-negativity-constrained least squares and the active set method," *SIAM J. on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 713–730, 2008.

[52] S. Arora, R. Ge, Y. Halpern, D. M. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu, "A practical algorithm for topic modeling with provable guarantees," in *ICML '13: Proc. of the 30th Int. Conf. on Machine Learning*, 2013.

[53] A. Kumar, V. Sindhwani, and P. Kambadur, "Fast conical hull algorithms for near-separable non-negative matrix factorization," in *ICML '13: Proc. of the 30th Int. Conf. on Machine Learning*, 2013.

[54] V. Bittorf, B. Recht, C. Re, and J. Tropp, "Factoring nonnegative matrices with linear programs," in *Advances in Neural Information Processing Systems 25*, ser. NIPS '12, 2012, pp. 1214–1222.

[55] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[56] N. Gillis, "The why and how of nonnegative matrix factorization," in *Regularization, Optimization, Kernels, and Support Vector Machines*, J. Suykens, M. Signoretto, and A. Argyriou, Eds., Chapman & Hall/CRC, 2014, ch. 12, pp. 257–291.

[57] D. Kuang, C. Ding, and H. Park, "Symmetric Nonnegative Matrix Factorization for Graph Clustering," in *Proceedings of the 2012 SIAM International Conference on Data Mining*, ser. Proceedings, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, Apr. 2012, pp. 106–117.

[58] R. L. Graham, D. E. Knuth, and O. Patashnik, “Concrete mathematics: A foundation for computer science,” in, 2nd. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994, ch. 2.2: Sums and Recurrences, p. 24.

[59] C. Ding, X. He, and H. D. Simon, “On the equivalence of nonnegative matrix factorization and spectral clustering,” in *in SIAM International Conference on Data Mining*, 2005.

[60] M. Girvan and M. E. J. Newman, “Community structure in social and biological networks,” *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, Nov. 2002.

[61] B. W. Kernighan and S. Lin, “An Efficient Heuristic Procedure for Partitioning Graphs,” *Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, Feb. 1970.

[62] S. Fortunato, “Community detection in graphs,” *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, Feb. 2010.

[63] G. Karypis and V. Kumar, “A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs,” *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, Dec. 1998.

[64] F. Pellegrini and J. Roman, “SCOTCH: A Software Package for Static Mapping by Dual Recursive Bipartitioning of Process and Architecture Graphs,” in *Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking*, ser. HPCN Europe 1996, London, UK, UK: Springer-Verlag, 1996, pp. 493–498.

[65] B. Hendrickson and R. Leland, “An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations,” *SIAM J. Sci. Comput.*, vol. 16, no. 2, pp. 452–469, Mar. 1995.

[66] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[67] I. Dhillon, Y. Guan, and B. Kulis, “Weighted Graph Cuts without Eigenvectors A Multilevel Approach,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[68] I. S. Dhillon, Y. Guan, and B. Kulis, “Kernel K-means: Spectral Clustering and Normalized Cuts,” in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’04, New York, NY, USA: ACM, 2004, pp. 551–556.

[69] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Physical Review E*, vol. 69, no. 2, p. 026 113, Feb. 2004.

[70] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey, “High Quality, Scalable and Parallel Community Detection for Large Real Graphs,” in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW ’14, New York, NY, USA: ACM, 2014, pp. 225–236.

[71] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, P10008, Oct. 2008.

[72] P. Pons and M. Latapy, “Computing Communities in Large Networks Using Random Walks,” *Journal of Graph Algorithms and Applications*, vol. 10, no. 2, pp. 191–218, 2006.

[73] M. Rosvall and C. T. Bergstrom, “Maps of random walks on complex networks reveal community structure,” *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, Jan. 2008.

[74] J. Baumes, M. Goldberg, M. Krishnamoorty, M. Magdon-Ismail, and N. Preston, “Finding Communities by Clustering a Graph into Overlapping Subgraphs,” in *Proceedings of the IADIS International Conference on Applied Computing*, N. Guimaraes and P. Isaias, Eds., ser. Applied computing, vol. 1, Lisbon, Portugal: IADIS, 2005, pp. 97–104.

[75] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, “Uncovering the overlapping community structure of complex networks in nature and society,” *Nature*, vol. 435, no. 7043, pp. 814–818, Jun. 2005.

[76] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, “Finding Statistically Significant Communities in Networks,” *PLoS ONE*, vol. 6, no. 4, e18961, Apr. 2011.

[77] J. Yang and J. Leskovec, “Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach,” in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM ’13, New York, NY, USA: ACM, 2013, pp. 587–596.

[78] ——, “Structure and Overlaps of Ground-Truth Communities in Networks,” *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 2, 26:1–26:35, Apr. 2014.

[79] J. J. Whang, D. F. Gleich, and I. S. Dhillon, “Overlapping community detection using seed set expansion,” in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, ser. CIKM ’13, New York, NY, USA: ACM, 2013, pp. 2099–2108.

[80] ——, "Overlapping Community Detection Using Neighborhood-Inflated Seed Expansion," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 5, pp. 1272–1284, May 2016.

[81] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection*, `http://snap.stanford.edu/data`, Jun. 2014.

[82] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '07, New York, NY, USA: ACM, 2007, pp. 29–42.

[83] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group Formation in Large Social Networks: Membership, Growth, and Evolution," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06, New York, NY, USA: ACM, 2006, pp. 44–54.

[84] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The Dynamics of Viral Marketing," *ACM Trans. Web*, vol. 1, no. 1, May 2007.

[85] dblp, *What is dblp?* `http://dblp.uni-trier.de/faq/What+is+dblp.html`, Accessed: 2015-06-29, 2015.

[86] ——, *Dblp: Computer science bibliography*, `http://dblp.uni-trier.de/`, Accessed: 2015-06-17, 2015.

[87] ——, *What do i find in dblp.xml?* `http://dblp.uni-trier.de/faq/What+do+I+find+in+dblp+xml.html`, Accessed: 2015-06-30, 2015.

[88] D. Zhou, J. Huang, and B. Schölkopf, "Learning with Hypergraphs: Clustering, Classification, and Embedding," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds., MIT Press, 2007, pp. 1601–1608.

[89] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Frontiers of Mathematics in China*, vol. 7, no. 2, pp. 365–384, 2012.

[90] C. Bothorel, J. D. Cruz, M. Magnani, and B. Micenková, "Clustering attributed graphs: Models, measures and methods," *Network Science*, vol. 3, no. 03, pp. 408–444, Sep. 2015.

[91] D. A. Cohn and T. Hofmann, "The Missing Link - A Probabilistic Model of Document Content and Hypertext Connectivity," in *Advances in Neural Information*

Processing Systems 13, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., MIT Press, 2001, pp. 430–436.

[92] E. Erosheva, S. Fienberg, and J. Lafferty, "Mixed-membership models of scientific publications," *Proceedings of the National Academy of Sciences*, vol. 101, no. suppl 1, pp. 5220–5227, Jun. 2004.

[93] R. M. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen, "Joint Latent Topic Models for Text and Citations," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08, New York, NY, USA: ACM, 2008, pp. 542–550.

[94] Y. Liu, A. Niculescu-Mizil, and W. Gryc, "Topic-link LDA: Joint Models of Topic and Author Community," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09, New York, NY, USA: ACM, 2009, pp. 665–672.

[95] A. Gruber, M. Rosen-Zvi, and Y. Weiss, "Latent Topic Models for Hypertext," in *Proceedings of the Twenty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, Corvallis, Oregon: AUAI Press, 2008, pp. 230–239.

[96] J. Chang and D. M. Blei, "HIERARCHICAL RELATIONAL MODELS FOR DOCUMENT NETWORKS," *The Annals of Applied Statistics*, vol. 4, no. 1, pp. 124–150, Mar. 2010.

[97] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A Model-based Approach to Attributed Graph Clustering," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12, New York, NY, USA: ACM, 2012, pp. 505–516.

[98] ——, "GBAGC: A General Bayesian Framework for Attributed Graph Clustering," *ACM Trans. Knowl. Discov. Data*, vol. 9, no. 1, 5:1–5:43, Aug. 2014.

[99] J. Yang, J. McAuley, and J. Leskovec, "Community Detection in Networks with Node Attributes," in *2013 IEEE 13th International Conference on Data Mining (ICDM)*, Dec. 2013, pp. 1151–1156.

[100] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining Link and Content for Community Detection: A Discriminative Approach," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09, New York, NY, USA: ACM, 2009, pp. 927–936.

[101] Q. Mei, D. Cai, D. Zhang, and C. Zhai, "Topic Modeling with Network Regularization," in *Proceedings of the 17th International Conference on World Wide Web*, ser. WWW '08, New York, NY, USA: ACM, 2008, pp. 101–110.

[102] Y. Sun, C. C. Aggarwal, and J. Han, "Relation Strength-aware Clustering of Heterogeneous Information Networks with Incomplete Attributes," *Proc. VLDB Endow.*, vol. 5, no. 5, pp. 394–405, Jan. 2012.

[103] M. Ester, R. Ge, B. Gao, Z. Hu, and B. Ben-Moshe, "Joint Cluster Analysis of Attribute Data and Relationship Data: The Connected k-Center Problem," in *Proceedings of the 2006 SIAM International Conference on Data Mining*, ser. Proceedings, Society for Industrial and Applied Mathematics, Apr. 2006, pp. 246–257.

[104] Y. Zhou, H. Cheng, and J. X. Yu, "Graph Clustering Based on Structural/Attribute Similarities," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 718–729, Aug. 2009.

[105] Y. Zhou, H. Cheng, and J. Yu, "Clustering Large Attributed Graphs: An Efficient Incremental Approach," in *2010 IEEE 10th International Conference on Data Mining (ICDM)*, Dec. 2010, pp. 689–698.

[106] Y. Ruan, D. Fuhry, and S. Parthasarathy, "Efficient Community Detection in Large Networks Using Content and Links," in *Proceedings of the 22Nd International Conference on World Wide Web*, ser. WWW '13, Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2013, pp. 1089–1098.

[107] J. Tang, X. Wang, and H. Liu, "Integrating Social Media Data for Community Detection," in *Proceedings of the 2011 International Conference on Modeling and Mining Ubiquitous Social Media*, ser. MSM'11, Berlin, Heidelberg: Springer-Verlag, 2012, pp. 1–20.

[108] J. Cruz, C. Bothorel, and F. Poulet, "Entropy based community detection in augmented social networks," in *2011 International Conference on Computational Aspects of Social Networks (CASoN)*, Oct. 2011, pp. 163–168.

[109] A. Strehl and J. Ghosh, "Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, Mar. 2003.

[110] H. Elhadi and G. Agam, "Structure and Attributes Community Detection: Comparative Analysis of Composite, Ensemble and Selection Methods," in *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, ser. SNAKDD '13, New York, NY, USA: ACM, 2013, 10:1–10:7.

[111] G.-J. Qi, C. Aggarwal, and T. Huang, "Community Detection with Edge Content in Social Media Networks," in *2012 IEEE 28th International Conference on Data Engineering (ICDE)*, Apr. 2012, pp. 534–545.

[112] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-View Clustering via Joint Nonnegative Matrix Factorization," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, ser. Proceedings, Society for Industrial and Applied Mathematics, May 2013, pp. 252–260.

[113] D. Jin, B. Gabrys, and J. Dang, "Combined node and link partitions method for finding overlapping communities in complex networks," *Scientific Reports*, vol. 5, Feb. 2015.

[114] R. Kannan, M. Ishteva, and H. Park, "Bounded matrix factorization for recommender system," in *Knowledge and Information Systems, 39(3):491-511*, 2014.

[115] R. Kannan, M. Ishteva, B. Drake, and H. Park, "Bounded matrix low rank approximation," in *Non-negative Matrix Factorisation Techniques: Advances in Theory and Applications, Ed. G.R. Naik, Springer Berlin Heidelberg, pp. 89-118*, 2016.

[116] X. Wang, L. Tang, H. Gao, and H. Liu, "Discovering Overlapping Groups in Social Media," in *2010 IEEE International Conference on Data Mining*, Dec. 2010, pp. 569–578.

[117] X. Wang, L. Tang, H. Liu, and L. Wang, "Learning with multi-resolution overlapping communities," *Knowledge and Information Systems*, vol. 36, no. 2, pp. 517–535, Aug. 2013.